Virtual Local Area Network

Based

Advance Driver Assistant System

Thesis submitted in partial fulfilment of the requirements for the award of degree of

Master of Technology

in

Computer Science & Applications

Submitted by

Aashima Sharma

(Roll No.-601534002)

Under the Supervision of

Dr. Maninder Singh

Professor & Head (CSED)

Mamta Bhatt

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA-147004

July 2017

CERTIFICATE

I hereby certify that the work is being presented in the thesis entitled, "Virtual Local Area Network based Advance Driver Assistant System", in the partial fulfilment of the requirements for the award of the degree of Master of Technology in *Computer Science and Applications* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Mamta Bhatt and Dr Maninder Singh and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not submitted for award of any degree of this or any other university.

(Aashima Sharma)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Dr. Maninder Singh

Professor & Head

CSED

ACKNOWLEDGEMENT

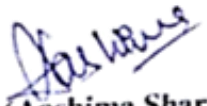
I would like to express my sincere gratitude to my mentors and supervisors Mamta Bhatt, Mr. Sundar and Mr. Ponnarasu for their immense help, guidance, stimulating suggestion and full time encouragement. They always provided me a motivational and enthusiastic atmosphere to work with. It was a great pleasure to do my internship under their supervision.

I am also thankful to Dr. Maninder Singh, Head, Computer Science & Engineering Department for his constant support and encouragement.

I would like to thank all the faculty members and staff of the department who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of this work.

I express my thanks to my family for their love, support and enthusiastic encouragement without which I could not complete this dissertation. I would like to thank to all my friends who helped me in all possible ways towards the completion of my work.

Finally I thank the Almighty who gave me the strength to complete the work.



(Aashima Sharma)

601534002

ABSTRACT

The automotive industry is rapidly expanding thus accelerating the demand for faster and additionally versatile network communication technologies. The automotive industry's future patterns like autonomous driving or Advanced Driver Assistance System (ADAS) require larger bandwidth to handle immense flow of data and strongly rely on well-timed connection and communication. Simultaneously, further high demands for future inter-ECU communication are becoming prominent and are expected to extend in near future. Similar considerations likewise hold for communication between vehicles and the external world, due to applications such as remote monitoring, fleet management, Internet-based automotive applications and Car-to-X communications.

Ethernet is a promising contender for in-car communications as an adaptable and scalable in-car network technology. The fundamental inspiration for this is the higher bandwidth provided by Ethernet (100Mbps onwards) when contrasted with current in-car networks which are quite less. It paves the way for applications like; ADAS that create the volume of exchanged data in automotive communication grow continuously; however the growing use of automotive Ethernet adds to a complicated hybrid network that involves varied security concerns. Consequently it poses new prerequisites for the communication protocols to cater such concerns. Thus, by discussing automotive Ethernet attributes with respect to the adaption of already existing security mechanisms such as VLAN we can overcome the assorted concerns of security in driver assistant applications.

Virtual Local Area Network (VLAN) technology segments a physical local area network (LAN) into several separate LAN's known as VLAN's. This technology not solely enhances network security in the LAN however also controls network broadcast domain traffic. It will permit ECU's on the same logical partition to communicate with one another whereas blocking others from causing the network traffic.

As already said that Ethernet is a promising candidate for in-car networks to fulfil the bandwidth and timing requirements but it also comes with some security concerns as well. Hence this work is proposed and VLAN is being implemented along with LwIP stack that is fundamentally an open source TCP/IP stack specifically designed for embedded systems.

.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SCREENSHOTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Automotive electronics is a sub-system consisting mainly of semiconductor devices used to sense, compute and actuate the different features/functions in a car. Using the next generation automotive microcontroller platform, automotive developers will be able to control power train and safety applications with one single MCU platform. Up to 300% performance surplus allows for more functionality and sufficient resource buffer for future requirements, keeping the power consumption on single core microcontroller level. While protecting IP, theft and fraud the automotive microcontroller provides an already built-in Hardware Security Module.

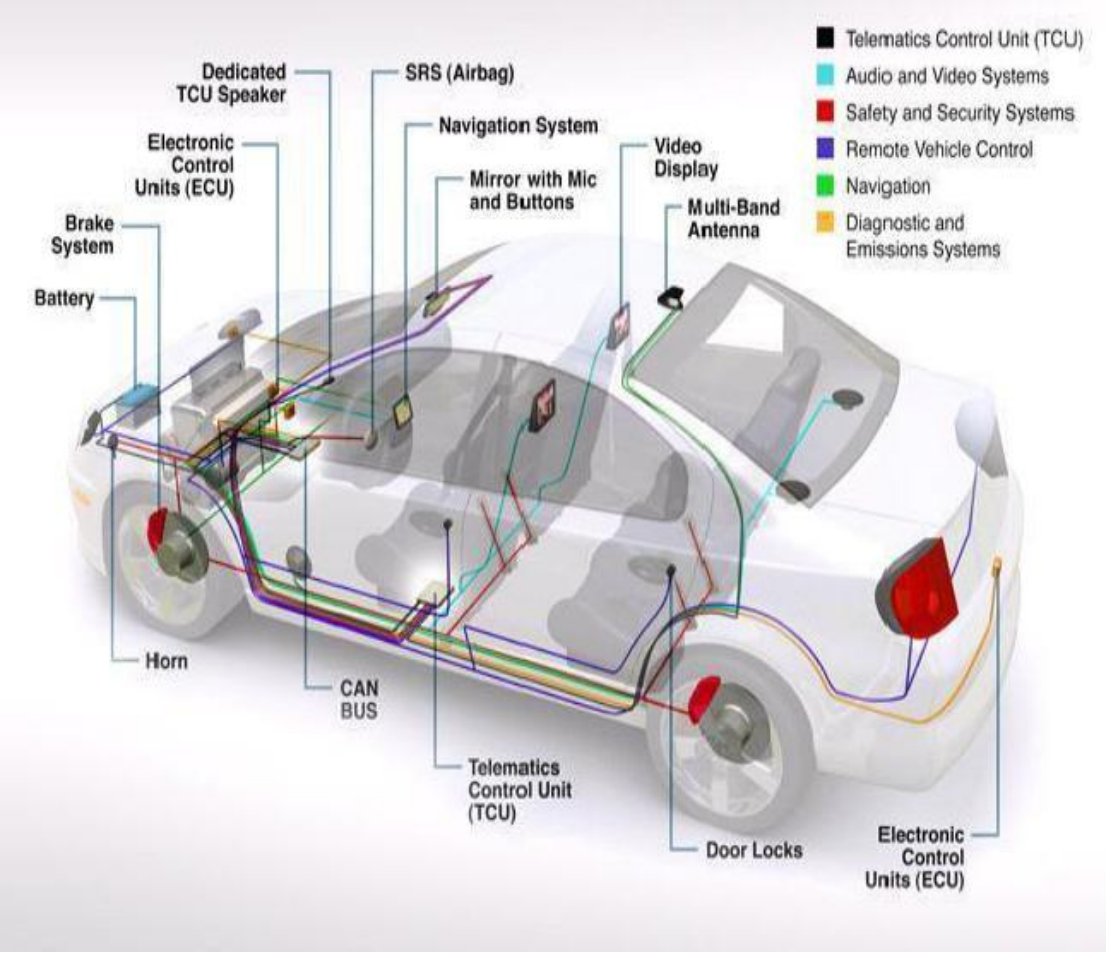


Fig 1: Major Electronic Control Units in Automobile

The impact of semiconductors on the automotive industry made mechanical and hydraulic components in vehicles to be replaced by electronic components since the 3970's.The low emissions, good performance and excellent drivability achieved by modern engines is made possible almost entirely by the monitoring, processing and real-time control capabilities of microcontroller(MCU) based electronic modules(ECU's, Electronic Control Units). Clearly, the availability of more optimized, better performing ICs and other semiconductor chips is the major factor pushing automotive technology today. Now a days there are more than 300 ECU's in modern luxury cars and up to 50 ECU's in commercial vehicles. For establishing communication among the different ECU's communication protocols such as CAN, LIN, Multi-CAN+, Flex-Ray etc. are implemented depending on the application needs.

1.1 AUTOSAR Software

AUTOSAR (Automotive Open System Architecture) is open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers. It is a partnership of automotive OEMs, suppliers and tool vendors whose objective is to create and establish open standards for automotive architectures that will provide a basic infrastructure to assist with developing vehicular software, user interfaces and management for all application domains. This includes the standardization of basic systems functions, scalability to different vehicle and platform variants, transferability throughout the network, and integration from multiple suppliers, maintainability throughout the entire product life-cycle and software updates and upgrades over the vehicle's lifetime as some of the key goals. This is the baseline for enabling a transition from an ECU specific software development to an application oriented approach.

Infineon provides MC-ISAR low-level drivers based on the AUTOSAR MCAL layer. With the MC-ISAR AUTOSAR drivers a system supplier can use one set of standardized basic software drivers over different applications within one configuration tool. By developing MC-ISAR in house at Infineon this enables efficient and optimized drivers. The complete AUTOSAR suite is provided in close cooperation with software partners which allow reusing their long term software experience. Thus an optimized AUTOSAR software bundle is available.

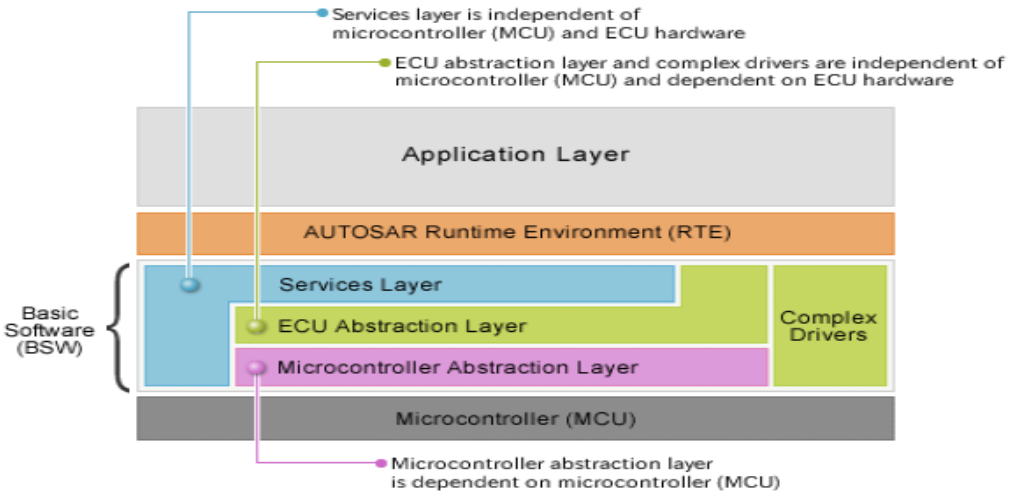


Fig 1.1: AUTOSAR Layered Architecture

1.2 MCAL (Micro Controller Abstraction Layer)

MCAL is a software module that directly accesses on-chip MCU peripheral modules and external devices that are mapped to memory, and makes the upper software layer independent of the MCU. Access to the microcontroller hardware is routed through the Microcontroller Abstraction Layer. The MCAL layer ensures a standard interface and controls the microcontroller peripherals. Details of the MCAL software module are shown below.

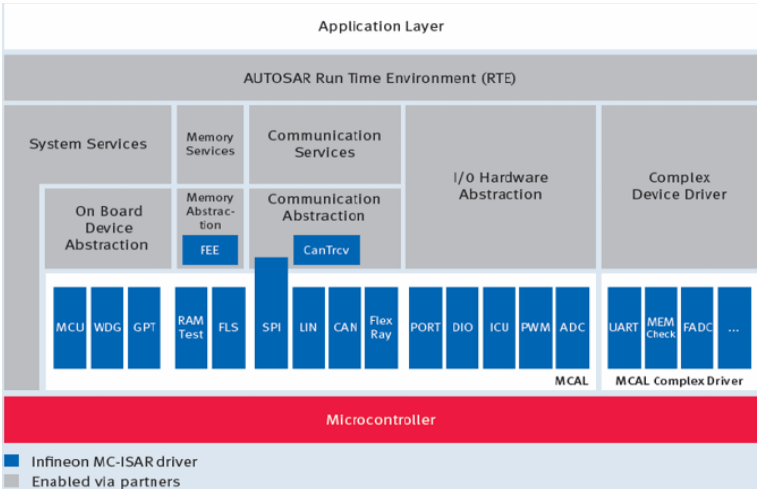


Fig 1.2: Details of the MCAL software module

1.3 System Reference

1.3.1 AURIX Microcontroller

AURIX (Automotive Real-time Integrated Next Generation Architecture) is Infineon's family of microcontrollers serving exactly the needs of the automotive industry in terms of performance and safety. Its innovative multicore architecture, based on up to three independent 32-bit TriCore CPUs, has been designed to meet the highest safety standards while increasing the performance at the same time. TriCore doesn't mean that it has three cores rather it means that it has three functional units such as RISC; DSP and microcontroller .These three components constitute TriCore. AURIX is a multicore controller family which usually has maximum of three TriCore's within it. The TriCore controller is designed to meet the needs of the most demanding embedded control systems applications where the competing issues of Price/performance, real-time responsiveness, computational power, data bandwidth, and power consumption are key design elements. With its high real-time performance, embedded safety and security features the AURIX family is the ideal platform for a wide range of automotive applications such as the control of combustion engines, electrical and hybrid vehicles, transmission control units, chassis domains, braking systems, electrical power steering systems, airbags and advanced driver assistance systems. In addition, the architecture used for AURIX allows a significant reduction in workload to develop safety systems compliant with today's highest Automotive Safety Integrity Level, the ASIL D standard.

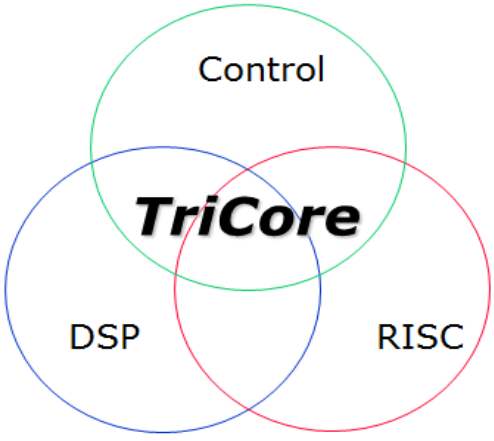


Fig 1.3.1: TriCore as combination of RISC, PCP and DSP

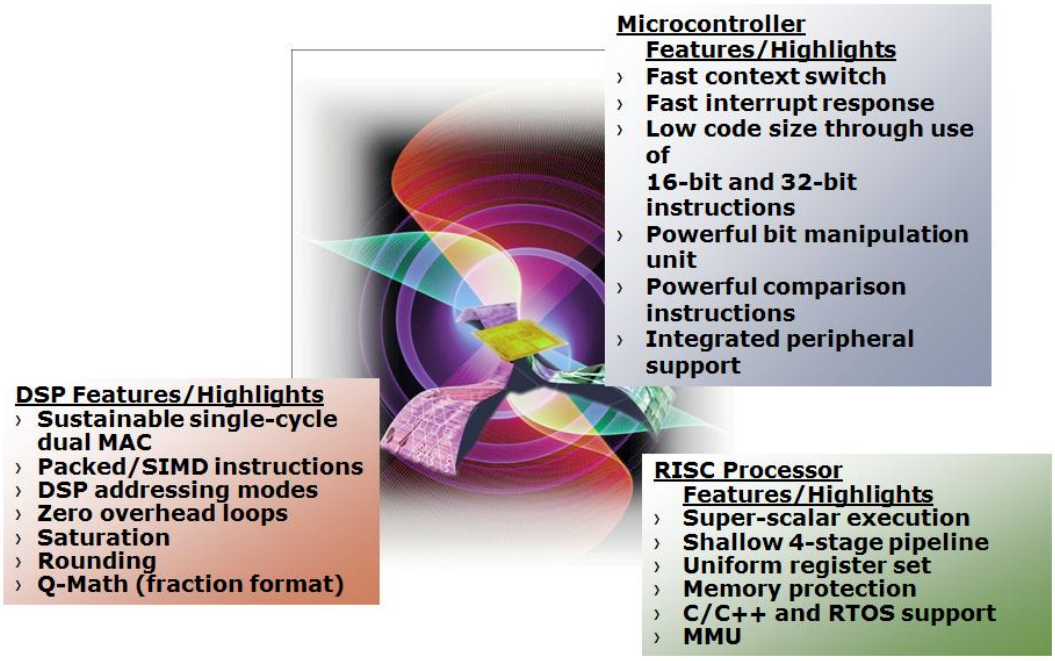


Fig 1.3.2: Detailed functionality of RISC, PCP and DSP used in TriCore

The TriCore has 32 general purpose registers which combines 36 data registers and 36 address registers, together makes lower context (D0 to D7 and A2 to A7) and upper context (D8 to D35 and A8 to A35). A0, A3, A8 and A9 are not classified into lower or upper context and are called global registers. TriCore also supports 64 bit data by using register pairs Address Register pair (P) and Data Register pair (E) as shown in the figure. Other than general purpose registers TriCore also have core registers used for handling different events occurring in CPU as shown in the figure:

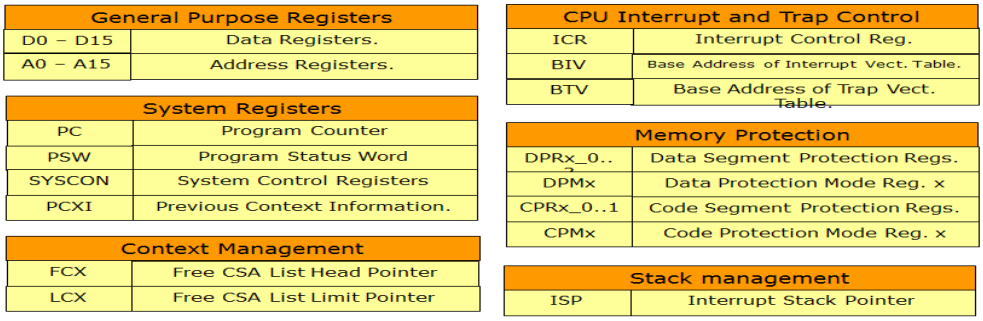


Fig 1.3.3: Core Registers TriCore

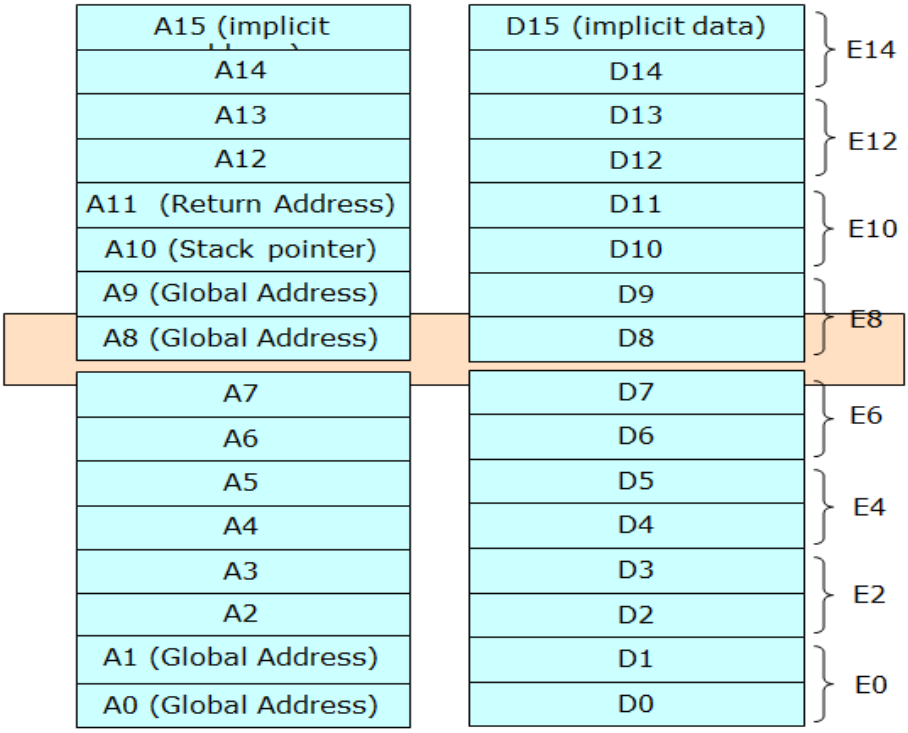


Fig 1.3.4: General Purpose Registers TriCore

1.3.2 Features of TC27x

Some of the main features of TC27X are mentioned as follows:

- 32-bit architecture
- 4 GB of address space
- 36-bit and 32-bit instructions for reduced code size
- Most instructions executed in one cycle
- Branch instructions (using branch prediction)
- Low interrupt latency with fast automatic context switch using wide pathway to on chip memory
- Zero overhead loop capabilities
- Dual, single-clock-cycle, 36x36-bit multiply-accumulate unit (with optional saturation
- Little-endian byte ordering for data memory and CPU registers
- Optional Floating-Point Unit (FPU) and Memory Management Unit (MMU)
- Extensive bit handling capabilities

- Single Instruction Multiple Data (SIMD) packed data operations (2x36-bit or 4x 8-bit operands)
- Flexible interrupt prioritization scheme Byte and bit addressing
- Memory protection
- Debug support

1.3.3 Applications

AURIX microcontroller has been used extensively in Power train, Safety and Advance Drive Assistance Systems (ADAS).

- Power train:
    - Engine Management Systems (EMS)
    - Transmission Management Systems (TMS)
    - Battery Management
- Safety:
    - Braking Systems
    - Electronic Power Steering (EPS)
- Driver Assistance Systems:
    - Radar Systems Camera
    - Based Systems

As Ethernet is being implemented with VLAN along with LwIP stack that is fundamentally an open source TCP/IP. So, this implementation on hardware front is based on AURIX microcontroller which is Infineon's family of microcontrollers serving exactly the needs of the automotive industry in terms of performance and safety and is implemented on various boards as per the demands of the OEM's. In this work TC27X board is being utilized for which the applications and features have been said above. Further, the MCAL layer which is a software module ensures a standard interface and controls the microcontroller peripherals. Access to the microcontroller hardware is routed through the MCAL layer. Additionally for this implementation AUTOSAR standard architecture has been followed. Hence providing a more secure environment for communication within in the car.

Chapter 2

LITERATURE SURVEY AND GAPS IN LIETRATURE

This literature survey focuses on the research done in the field of automotive in-vehicle security. Most of the work proves requirement of security in the automotive domain and its future potential and necessity. The existing vulnerabilities in the in-vehicular networks, and current approaches to mitigate these problems:

2.1 LITERATURE SURVEY

In this paper [1], the author proposes that the use of ethernet will bring a lot of changes and will also support reuse of various components and tools inside the car. It will act as an enabling technology for various advanced features in the automotive domain. And Ethernet can also effectively handle a large amount of data transfer from vehicle to vehicle and vehicle to the outside world. The author also says that ethernet can be an enabling link between the vehicle connectivity and internet which can open doors for IOT (internet of things) in an automotive domain as well. This has been reflected in the paper that automotive ethernet will bring fundamental benefits as it will provide scalability and flexibility for next generation network architecture, will provide increased communication bandwidth and hierarchical network topologies. As automotive ethernet will bring a lot of changes while, CAN, Flex ray will remain prominent for safety critical communications only. Also it proposes that some work is still required to validate the existence of different data classes on the same network and also to develop new automotive optimized components for ethernet networks.

In this paper [2], presents the protocol for on-board and off-board communication. Because of limited resources in embedded systems light weight internet protocol is being used which is basically a TCP/IP implementation being adopted. The author presents the views for using IP in the automotive family because of the need for communication between the onboard equipments and external test equipments. Outline of DoIP, Ethernet innovation and lightweight TCP/IP are talked about in this paper. The Autosar standard empowers the utilization of a part based programming configuration show for the outline of a vehicular framework. It reflects how sooner rather than later, Ethernet innovation turns into the more alluring ones to give high data transfer rate for the communication

between in-vehicle ECUs. The asset confinements of the inserted hardware drive the improvement of lightweight TCP/IP for successful correspondence.

This paper [3], presents how ethernet is a designated as CAN successor because of its high throughput and because of its capability to reuse the standard technologies like TCP/IP for better integration. This illustrative application made cement the probability to utilize a high throughput organize as future system for in-vehicle correspondence, keeping up in reverse similarity with existing system. It reflects that it is important to develop and execute a synchronization amongst Ethernet and CAN systems. A future implementation can be planned and figured it out concentrating on data transfers. At last the utilized standard will allow better joining with outer world, similar to remote joining with specially appointed systems for agreeable works for vehicles that help this standard. In this future change special care will be taken to exhibitions, making an arrangement of estimations to assess the results that could be attained using different methodologies.

This paper [4], proposes that because of increasing demands of bandwidth in vehicles because driver assistant, infotainment and entertainment ethernet can prove to be a great candidate to fulfil all the required demands. Real time ethernet is a scalable approach to reduce the complexity of in-vehicle communication structure. It proposes a RT ethernet prototype based on ARM to achieve the timing and bandwidth characteristics of future automotive application which is basically based on advanced interrupt driven architecture. This paper provides analysis on how the ethernet prototypes are suitable for safety critical environment. It shows that how ethernet is a potential candidate for in car networks because of the importance gained by the ECU's for research and industry. They have implemented this prototype for camera based driven assistance systems. But on the other hand it also analysis for applications with high demand on computing power.

As safety is the most important concern of automotive systems which is further being enhanced by Advanced Driver Assistance Systems (ADAS) and other advanced autonomous functions, and as a result automotive electronic systems are becoming more distributed and connected than ever. And these connections become grounds for various security attacks. In this paper [5], we address safety and security together, because Ethernet-based automotive networks which are believed to be the next-generation automotive networks .This paper discusses interactions between safety and security in frame replication and elimination and VLAN. But there is a major trade off between

safety and security with VLAN segmentation as security is a necessary condition for safety from functional perspective. For automotive systems, Ethernet-based networks provide the platforms for considering safety and security together. Future directions include low-overhead security protection safety and security modelling and abstraction, simultaneous safety and security optimization and trade-off analysis.

Nowadays we see a trend in the past few years that there has been a large increase in the complexity of electronic systems due to the increasing number of ECU's in the vehicle architecture. This paper [9], identifies which security mechanisms are already implemented, and which should be introduced in the future. It further discusses which parts of each mechanism require special hardware support. It also questions if the modern microprocessors for the automotive can cover all the requirements for the memory and I/O systems. Further it reflects that lightweight cryptography can also come forward as a major section to meet the various performance and other important perspectives of a modern vehicle which impose a significant impact on the costs.

On contrary it says that still a lot of work needs to be on various aspects related to standardization such that a common approach is followed.

As the demand for automotive ethernet is increasing in the near future because of the increasing bandwidth and timing demands. Thus, the author in this paper [10], focuses on discussing automotive Ethernet attributes with regard to the adaption of existing security mechanisms in contrast to the potential of creating new ones, and several challenges emerge in consideration because of comparatively fewer available resources and the integration into a vehicle environment. The significant qualities of Ethernet are its effortlessness and zero- configurability. These are additionally the reason for its shortcomings; the dynamic elements that enable it to self-design can be misused. Ethernet can be secured to a sensibly abnormal state by directing all switches, hosts, and clients halfway and applying cryptographic strategies. Nonetheless, this implies the loss of simplicity and zero-configurability. Likewise existing arrangements are genuinely granular and don't protect extremely well against misuse. This could almost be viewed as equivalent to the assurance given by supplanting switches with IP switches, aside from ARP communicates which leave the framework powerless against misuse. A few research approaches give the possibility to enhancing the security of Ethernet. Incorporated administration can be streamlined and made more granular.

This paper [11], talks about in-vehicle physical network architecture typically used in automotive and their characteristics such as cost, performance, their applications and certain limitations. Later it shows with no authentication and confidentiality it is trivial to attack these bus representations especially with more physical access. To decide appropriate security level this paper also analysis motivations and ability for different people to hack the vehicle. In this work, they have quickly exhibited present and future vehicular communication systems and pointed attention to a few bus communication security issues. They displayed an approach that makes the utilization of present day communication security mechanisms to settle the majority of the vehicular communication security issues. It reflects that multimedia buses and remote communication interfaces will soon be accessible in most present day cars. Since future car frameworks and plans of action especially rely upon complete and productive measures that give vehicular correspondence security, satisfactory specialized, and money related uses must be organized today.

As we know that the new upcoming driving assistance applications are posing new requirements on the communication protocols and so as to have a better understanding of these emerging requirements and their complexities the author in this paper [12], proposes the challenges of such systems using the side view system which uses five cameras for input as an example. This paper gives an approach of all best in class car correspondence frameworks which can be utilized for present day camera based driver assistance frameworks. The communication systems were exhibited for their value and suitability for current camera based driver assistance frameworks. In view of the decision of the communication systems which is reasonable for present day camera based driver help frameworks the setup of the Side-View framework was clarified. Moreover, two utilize cases were depicted which both utilize the Side-View framework. As subsequent stage it is required to enhance the framework with a new image processing algorithm.

In this paper [13], as the name suggests, it practically demonstrates the vulnerabilities of in-vehicle network and ability to amplify the attack to safety-critical systems. The authors showcased that they can control safety critical features by inserting malicious code to CAN network with various tests. Although their method required wired connection with CAN network which considered as infeasible for an intruder in practical life.

As automotive new trends made possible to connect the mobile phone and interact with the automobile, a new threat of privacy comes to light. This paper [14], proves that in in-

vehicle communication privacy can be at risk as the networks are not equipped with security mechanism and even this information can be used to invade car's integrity. The paper proposes use of security frameworks to monitor data flow between ECU to have elevated security and privacy.

The author in this paper [15], gives various facts and figures towards the introduction of ethernet in automotive communication and also discusses how ethernet will provide benefits to different automotive functional domain. It also discusses how ethernet acts as a possible candidate for automotive industry.

This paper [16], describes the evolutionary path of bringing ethernet in automotive industry focusing on electric mobility. The use of ethernet for in-vehicle connectivity can manage large chunks of data to the outside world and between vehicles –to-vehicle communication. It shows how ethernet is an enabling technology for introducing advanced features in automotive. It is clear that Ethernet speaks to be the promising contender for the cutting edge car systems. The advantages of a wide-scale appropriation of Ethernet are boundless and incorporate data transfer capacity upgrades, cost funds, and enhanced execution adaptability. Since Ethernet is a broadly utilized and perceived IEEE standard, the car industry will profit by it's proceeded with advancement in time. It is likely that the move toward completely Ethernet-based car systems will be developmental and not progressive. This paper has proved that, as in-vehicle innovation turns out to be increasingly perplexing, there is a drive to standardize approaches over the business, permitting makers to concentrate on advancing with energizing applications based on comparative establishments. This gives a brilliant system to the future development and change of in-vehicle frameworks, driving eventually to more driver comfort and, most importantly safety.

2.2 GAPS IN LITERATURE

1. In most of the cases when we are talking about in-car networks the work which is proposed is most of the times based on LIN, CAN, Flex ray but as the demand for bandwidth is increasing Ethernet comes into the picture. But again Ethernet has some of its own disadvantages as well because as the amount of data being exchanged increases the threats to the internal network of the automotive increase as well. Here security is one of the major factors required to be covered along

with Ethernet without posing any new requisites and to also overcome the security breaches.

2. When we talk about VLAN, a large amount of research exists in the conventional ethernet networks but when we talk about VLAN in automotive it poses some challenges to understand the existence and implementation of VLAN with respect to the traditional automotive networks.
3. Limited research exits for the LwIP stack in collaboration with Ethernet which is nowadays a new boom in automotive as it is specifically designed for embedded systems that provides with the basic advantage of reducing the resource usage while still having a full scale TCP stack.
4. Also when we talk about LwIP, dynamic memory allocation is also a major trade-off for its productive usage.

As a part of this work VLAN implementation on ADAS systems is being covered which is embedded with ethernet frame on a LwIP Stack.

CHAPTER 3

PROBLEM STATEMENT

Today's present time depends a lot on information exchange and communication. Automotive industry has evolved in similar fashion, to offer more on road safety and connectivity. Now, vehicles doesn't simply interacts with its driver, however conjointly with numerous other entities like other vehicles, infrastructure, servers on internet to receive or send information, that can also be vital information regarding cyber-physical features of a car for safety critical functions.

Such a system connects in-vehicle ECU network through vehicles central gateway for V2X communication to exchange required information. ECUs are connected in clusters and these clusters are connected with gateway to form an in-vehicle domain network of an automobile. A lot of information exchange between clusters happens to reduce number of required sensor. Each cluster of ECU belongs to a functional domain, like power train, chassis, body, and infotainment/telematics.

In such a scenario inter/intra-domain communication in car requires information security, as automotive network are vulnerable to cyber-attacks. One might launch attacks on ECU's hardware, software or networks to add anomaly and alter its safe state. If one bus ECU is breached attack can be amplified to alternative bus ECUs.

Nearly all cars can be vulnerable at hackers. There are a multiple of examples where we can see how vulnerable attacks were launched from external environment within the car network. One of the examples is mentioned here where a group of individuals disable SUV on a busy highway:

A person named Andy Greenberg [18] was driving along a busy street recently when he suddenly had no control of his vehicle. The accelerator abruptly stopped working. The car crawled to a stop. As 18-wheelers whizzed by his stalled vehicle, it had been hacked.

And to cater all these security concerns we will separate the public domain in a car from the private domain using a common communication protocol. It is nearly not possible to form a 100% secure communication network however we will perpetually have some live to guard it.

So here a technique is being proposed of tagging the Ethernet frame using a VLAN tag to separate the public domain in a car from the private domain thereby providing a more secure communication network in a car. Whereby we can restrict the ECU's from unapproved external access and unencrypted data. And a consequence of this implementation only the VLAN tagged frames will be allowed to enter the ECU's connected through Ethernet. There by if any external source intends to interface with the car remotely through infotainment or attempt's to hack into the ECU's through a false packet; then a VLAN tag is required for it to enter the system. And in the absence of this VLAN tag in the frame the access will be denied for that packet to enter the car framework. The false packets won't be able to do any reasonably harm with the inner network of the automobile. Hence Ethernet is being implemented with VLAN along with LwIP stack that is fundamentally an open source TCP/IP stack specifically designed for embedded systems that provides with the basic advantage of reducing the resource usage while still having a full scale TCP stack.

Objectives of thesis

The major objectives targeted for this work are mentioned as follows:

1. To insert the VLAN header into the ethernet frame.
2. To make the transmission and reception compatible with the LwIP stack and furthermore with the TC27x hardware.
3. To validate the ICMP, ARP, TCP, UDP, VLAN frame reception and transmission between the test bench application and the TC27x hardware.

CHAPTER 4

ETHERNET AND LwIP IN AUTOMOTIVE

4.1 INTRODUCTION

Ethernet is one of the emerging candidates for in-car communications. The main motivation for this is the higher bandwidth provided by Ethernet (100 Mbps onwards) as compared to current in-car networks. Such an increased bandwidth paves the way for applications, like Advance Driver Assistance Systems (ADASs), which make the volume of exchanged data in automotive communication grow continuously. In addition, Ethernet technology is scalable, thus meeting the scalability requirements imposed by today's automotive systems, where the number of nodes to interconnect steadily increases.

Another enabling factor for using Ethernet as a common networking technology for in-car communications is the accessed technology, which entails that there is a large knowledge already available that allows for better testing, maintenance and development.

Further traffic requirements are steadily growing so there is a need for more bandwidth. Premium cars today count more than 70 ECUs that implement hundreds of distributed functions to provide, e.g., comfort, safety, infotainment, and which produce many communication exchanges. The demand for inter-ECU communication has also heavily increased and is expected to grow in coming future. More bandwidth is also required by On-Board Diagnostics (OBD). The measure of software embedded in the cars today is increasingly growing, due to the steady advancements of functionalities provided by automotive electronic systems. OBD is used by many vehicle functions such as, emissions, monitoring, diagnosis of components and maintenance with the possibility of downloading and updating software.

Ethernet being a common network technology would reduce the communication complexity considerably. The communications are quite different from one functional domain to another. In some domains such as power train real-time communication is the main requirement while other domains may have relaxed time constraints but require more bandwidth

The communications may be quite different even within the same functional domain when several functions are present that feature different data rates and constraints. As a result, several network types are also found within the same domain, with different characteristics in terms of bandwidth, real-time support, etc. This difference is also reflected in the different network technologies adopted, from LIN used mainly for low-

speed communications in the body and comfort domain , to CAN used in various domains from bandwidth ranging from 100 to 500 Kbps and to Flex Ray that provides 10Mbps. Hence Ethernet provides suitable automotive communication.

4.1.1 ETHERNET NETWORK ELEMENTS

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

• **Data terminal equipment (DTE)**—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)**—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.

4.1.2 ETHERNET MAC SUBLAYER

The MAC sub layer has two primary responsibilities:

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

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

4.1.3 ETHERNET FRAME FORMAT WITH VLAN TAG

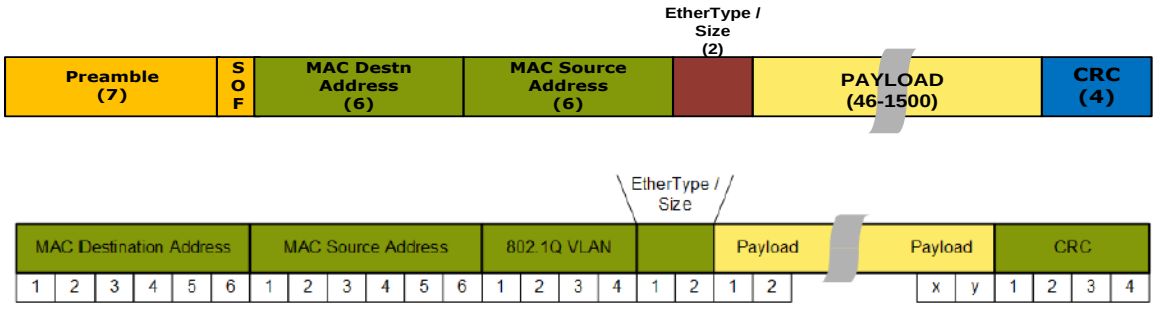


Fig 4.1.1: Ethernet frame with VLAN tag

The various fields of Ethernet frame format are:

- MAC Destination address – 6 bytes
- MAC Source address – 6 bytes
- VLAN Tag – 4 bytes
- Ether Type – 4 bytes
- Payload – 44 – 1500 bytes
- CRC – 4 bytes

ETHER TYPES: The various types of ethernet headers which are used are mentioned as follows which represent the various protocols.

ETHERNET HEADER	PROTOCOL
0x800	IPv4
0x806	ARP
0x8100	804.1Q VLAN Tag
0x86DD	IPv6
0x88F7	Precision Time Protocol IEEE 1588 over Ethernet

Table 4.1.3: List of ethernet headers

4.1.4 ETHERNET PROTOCOL STACK STRUCTURE & BASE TECHNOLOGIES

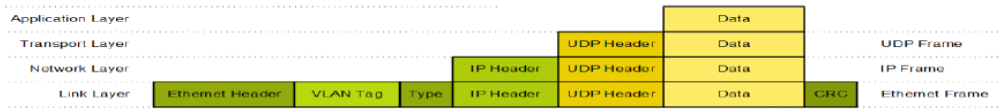


Fig 4.1.2: OSI stack structure

The various sections of the OSI stack which are given in the figure above are explained as follows:

- Ethernet header contains the MAC address of the packet.
- VLAN Tag contains the VLAN ID and PCP.
- Type tells Ether Type contained in the packet.
- IP header is the IPv4 or IPv6 header.
- UDP header can be TCP or UDP header.

The OSI stack at various layers is interpreted in the form of ethernet protocol structure in the implementation as described in the figure given below.

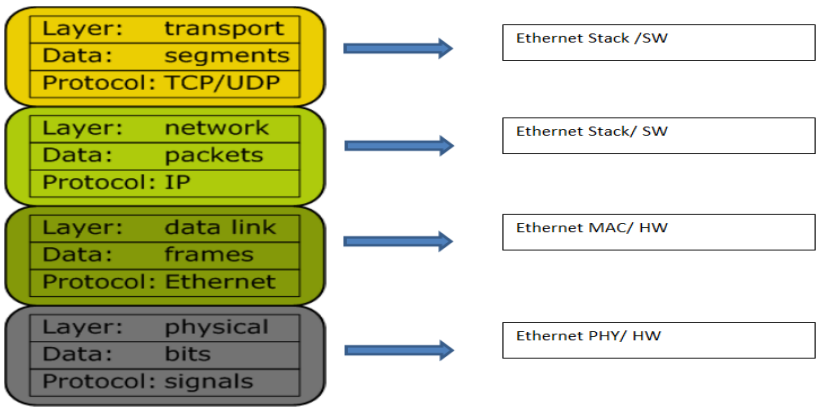


Fig 4.1.3: Ethernet protocol stack

In our implementation we have various layers which interface and communicate with each other and have a level of dependency with each other which is described as follows. Here IP stack is considered as the LwIP stack and the operating system is used is Erica (Infineon's internal operating system).

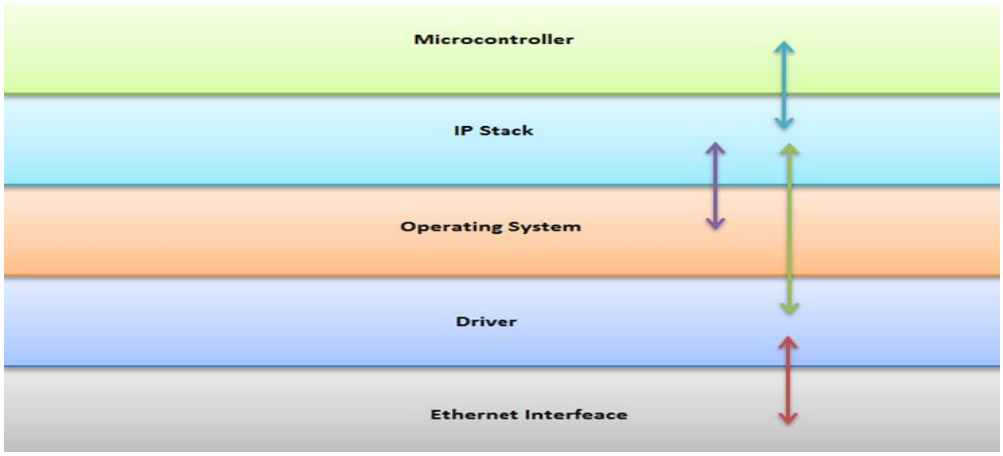


Fig 4.1.4.: Dependency overview of hardware and software component

4.2 AURIX ETHERNET OVERVIEW

4.2.1 KEY FEATURES: The main features of Aurix ethernet are mentioned as follows:

- Supports 10/100-Mbits data transfer
- IEEE 804.3 compliant RMII/MII interfaces
- Supports both full-duplex and half-duplex operation
- Preamble and start-of-frame data (SFD) insertion in Transmit, and deletion in Receive paths
- Automatic CRC and pad generation.
- Checksum Offload engine.
- Flexible Address filtering modes
- Complete network statistics with Mac Management Counters
- MDIO Master interface for PHY device configuration and management.
- Independent DMA block
- 4KB Tx/Rx FIFO buffers in MAC

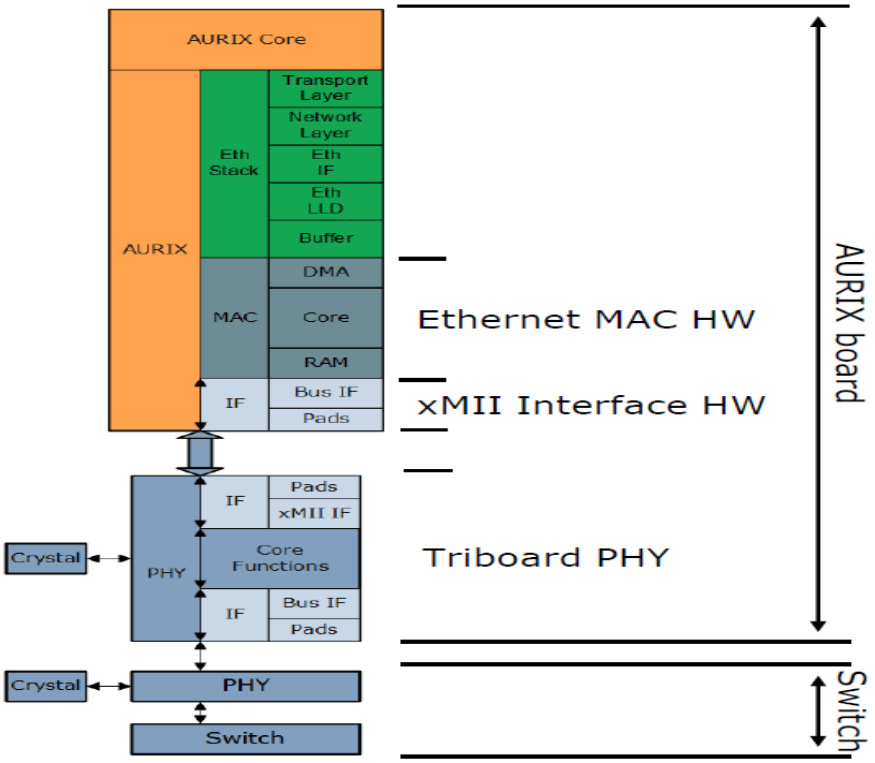


Fig 4.2.1: Aurix Ethernet Setup

This setup shows the interface between the Aurix core and the ethernet driver along with the hardware specifications. It shows where and how the microcontroller holds their position along with the RAM and the buffers. Also it shows the interface between the Aurix board and the xMII bus. It basically shows the hardware and software Aurix and ethernet setup.

4.2.2 ETHERNET DMA

The DMA has independent Transmit and Receive engines, and a CSR space. The Transmit Engine transfers data from system memory to the device port (MTL), while the Receive Engine transfers data from the device port to system memory. The controller utilizes descriptors to efficiently move data from source to destination with minimal Host CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the Host CPU for situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions. The DMA and the Host driver communicate through two data structures:

a) Control and Status registers (CSR)

b) Descriptor lists and data buffers

Controller Features of DMA:

- DMA has independent Transmit and Receive engines
- DMA Transmit Engine transfers data from system memory to the device port
- DMA Rx Engine transfers data frames received by the core to the Receive Buffer in the Host memory
- DMA Controller utilizes descriptors to efficiently move data from source to destination with minimal Host CPU intervention
- DMA controller can be programmed to interrupt the Host CPU on Tx/Rx completion. Two separate descriptor lists; one for reception, and one for transmission. DMA is the core of Ethernet MAC. Most important part. Configure DMA properly and you are done.

4.2.3 DMA DESCRIPTORS:

Descriptors are enablers for DMA operations. Descriptor architecture allows large blocks of data transfer with minimum CPU intervention. They contain all information to make DMA operate independently. They are configured by Driver software. Descriptors occupy user RAM memory. The various features of descriptors are:

- Descriptors contains the information about frame buffer address and length
- DMA stores the information about received and transmitted frames in Descriptors
- Descriptors should be initialized before any DMA transaction can take place.
- Descriptor is forward Linked List. 1st links to 4nd, 4nd to 3rd and so on.
- Last one links to 1st one to make a ring mode.
- Links can be explicit or implicit.
- DMA maintains the following:
    a. Base address of the Descriptor linked list
    b. Next Free Descriptor
    c. Last descriptor is marked, so that DMA rolls back to 1st descriptor

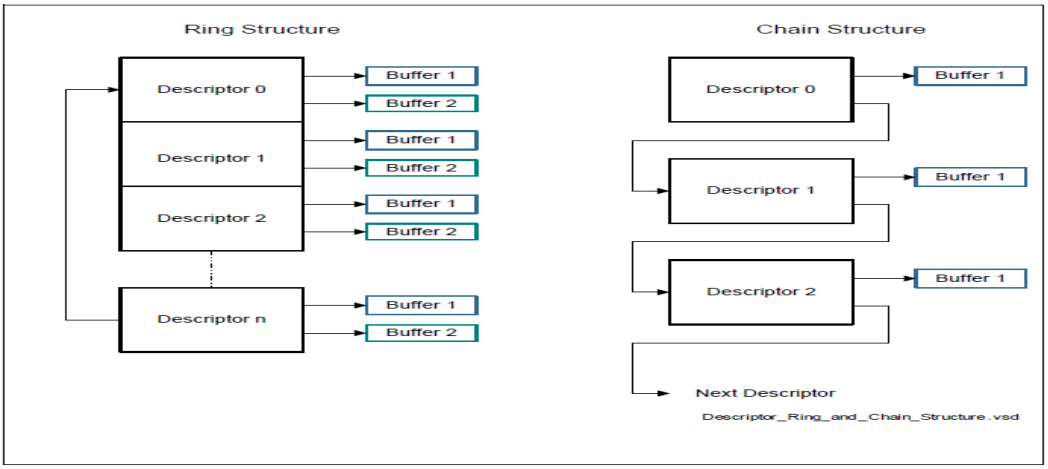


Fig 4.2.2: Structure supported for descriptors

DMA supports two kinds of structures i.e. Ring structure and Chain structure. The various features of Ring structure has been mentioned as follows:

- Descriptors build a ring:
- There is a ring per DMA controller
- Max 4 rings for Rx and 4 rings for Tx
- The ring size is not limited
- **Size of descriptors:** 34 bytes or 64 bytes. AURIX supports both.

The DMA transfers data frames received by the core to the Receive Buffer in the Host memory, and Transmit data frames from the Transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers. There are two descriptor lists; one for reception, and one for transmission. The base address of each list is written into DMA Registers 3 and 4, respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. The descriptor lists resides in the Host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory. The structure of receive and transmit descriptor are shown as follows:

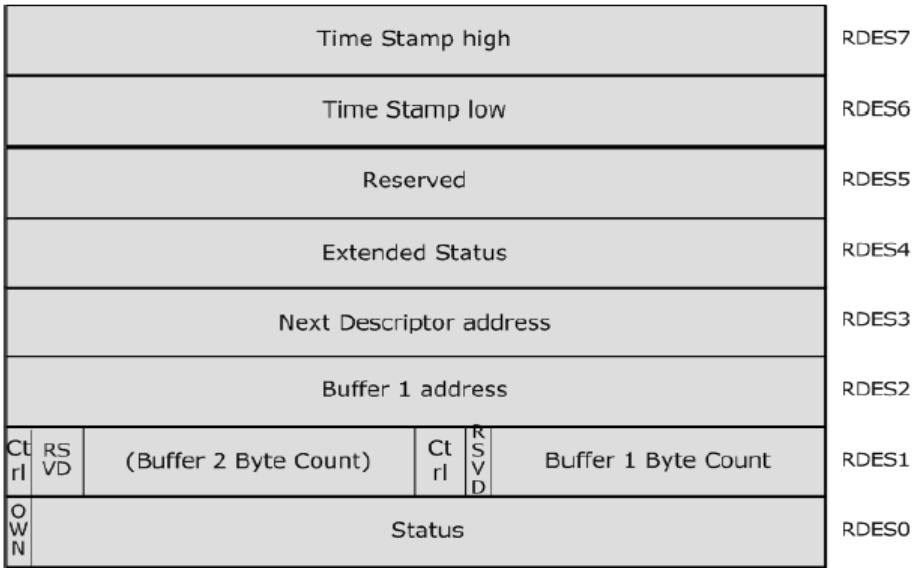


Fig 4.2.3: Structure of Receive descriptor

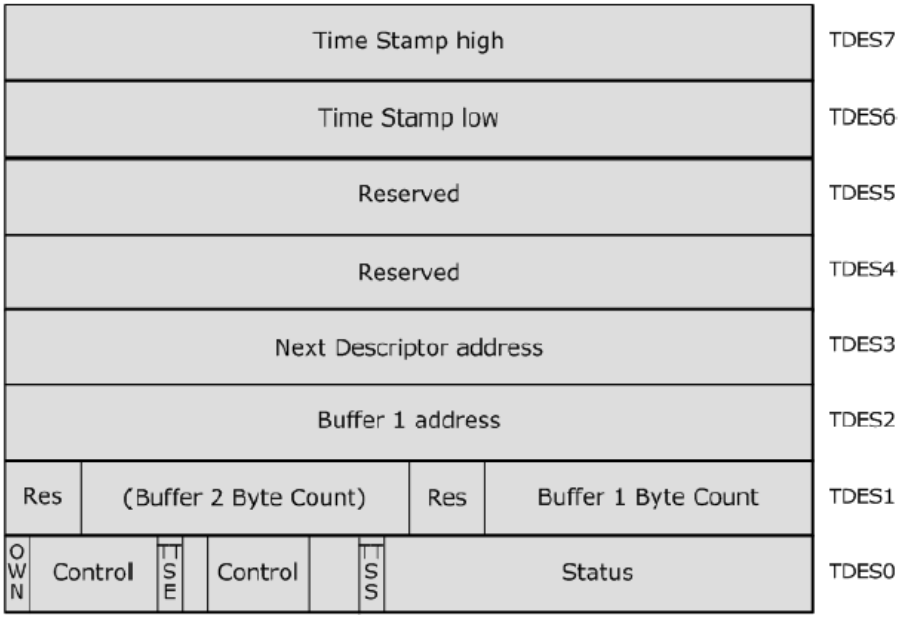


Fig 4.2.4: Structure of Transmit descriptor

4.3 ETHERNET Tx & Rx ACTIVITIES

The Ethernet transmit and receive activities have been explained below:

4.3.1 ETHERNET FRAME TRANSMISSION

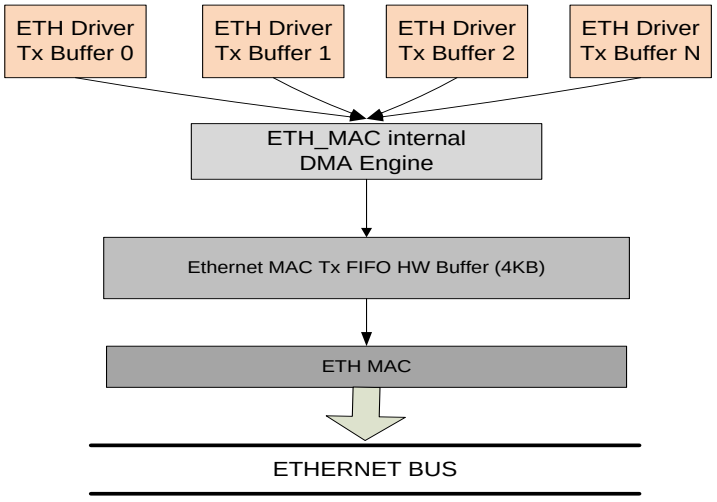


Fig 4.3.1: Ethernet frame transmission

- Number of Buffers are configurable
- Size of buffers is configurable.
- Size of buffers should be multiple of 4 bytes.
- Buffer Address should be 4 bytes aligned

4.3.1.1 Simple Tx Data Flow

Tx DMA 0 is triggered by application. Tx DMA 0 sets up transfer which is defined by "Descriptors". Tx DMA 0 transfers' data blocks to Tx Queue 0.Connection Tx DMA 0 to Tx Queue 0 is permanent.MAC reads Ethernet packet from Tx Queue 0.MAC inserts Tags. Ethernet packet is sent via Xmii

- Tx DMA closes packet move
- DMA writes status info
- DMA creates interrupts on demand
- DMA creates error messages (in case of errors)
- Tx DMA expects typically

- Complete packet with MAC DA & SA, VLAN, Ether Type
- Pad and CRC are added manually or by MAC
- Insertion of MAC Source Address and VLAN Tags is configurable

4.3.2 ETHERNET FRAME RECEPTION

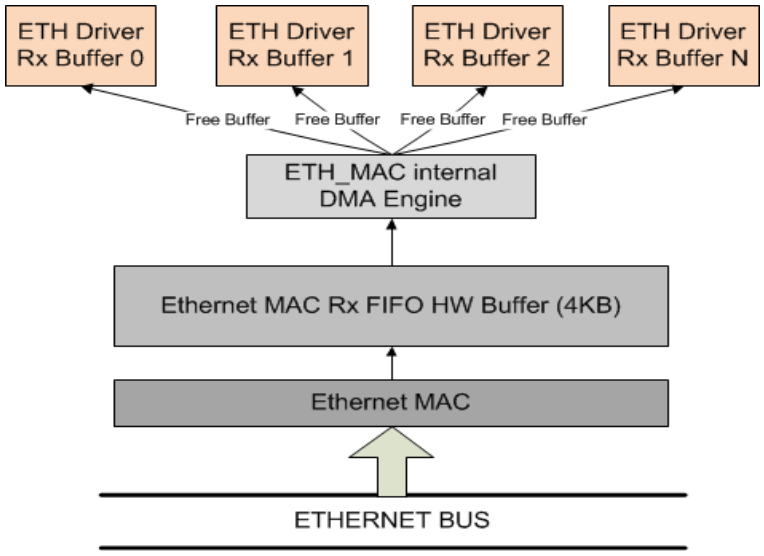


Fig4.3.2: Ethernet frame reception

- Number of Buffers are configurable
- Sizes of buffers are configurable.
- Size of buffers should be multiple of 4 bytes.
- Buffer Address should be 4 bytes aligned

4.3.2.1 Rx Packet processing

- Rx DMA always tries to allocate two descriptors in advance
- MAC only forwards packets which passed the address filters
- All filters can be switched off -> all packets received
- MAC removes packets < 64 byte

4.4 ETHERNET MCAL OVERVIEW AND FILE STRUCTURE

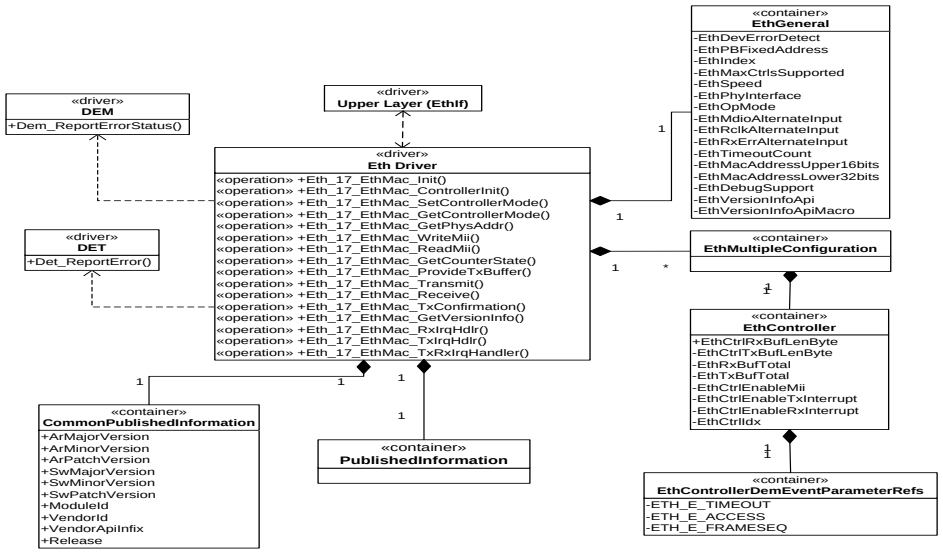


Fig 4.4.1: Ethernet MCAL overview

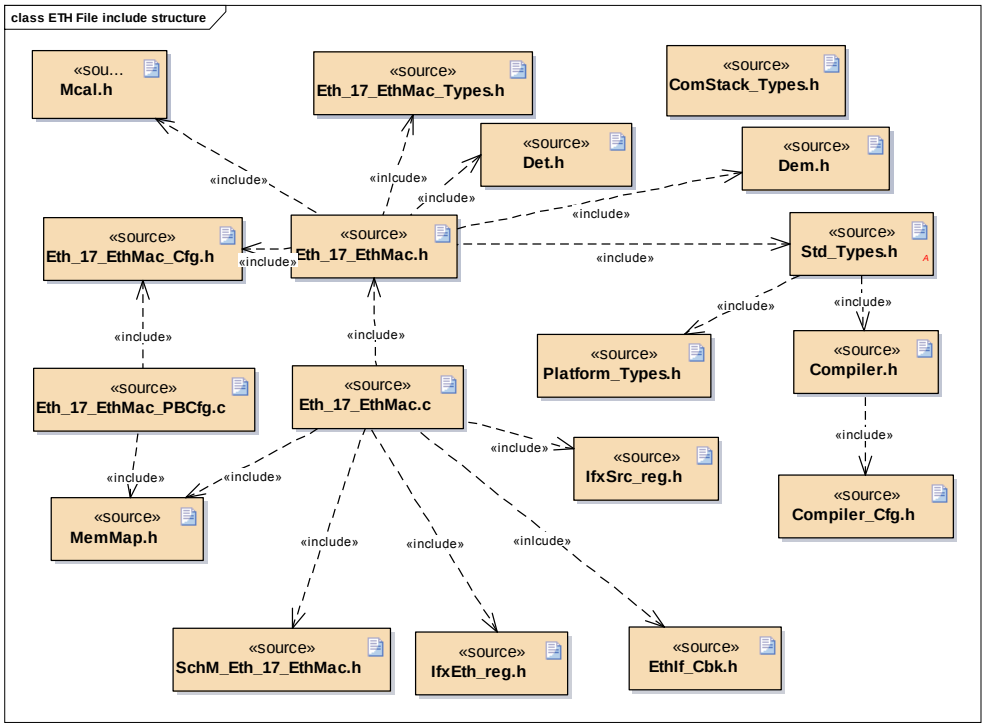


Fig4.4.2: Ethernet file structure

4.5 ETHERNET INITIALIZATION API'S

1. **Ethernet – Initialization API's – 1**

- void Eth_17_EthMac_Init(const Eth_17_EthMac_ConfigType* CfgPtr)
- **Functions:**
  i. Store information related to PB config root locally.
  ii. Change the state of the component from ETH_STATE_UNINIT to ETH_STATE_INIT.

2. **Ethernet – Initialization API's – 4**

- Std_ReturnType Eth_17_EthMac_ControllerInit(uint8 CtrlIdx, uint8 CfgIdx)
- **Functions:**
  i. Enable the MAC clock (CLC – clock control)
  ii. Configure the operation mode (100/10 Mbps, RMII/MII, ALT PIN config etc.)
  iii. Reset of MAC and DMA (All PHY clocks mandatory during SWR)
  iv. Interrupt Configuration
  v. Buffer and DMA descriptor initialization
  vi. Change the state of the component from ETH_STATE_INIT to ETH_STATE_ACTIVE.

3. **Ethernet – Initialization API's – 3**

- Std_ReturnTypeEth_17_EthMac_SetControllerMode(uint8CtrlIdx, Eth_ModeType CtrlMode)
- **Functions:**
  i. Set Controller to MODE_DOWN / MODE_ACTIVE.
  ii. MODE_DOWN: Disable Tx/Rx. Reset all Tx and Rx buffers. Reset Descriptors.
  iii. MODE_ACITVE: Enable Tx/Rx.

4. **Ethernet – Transmit API's - 1**

- BufReq_ReturnType Eth_17_EthMac_ProvideTxBuffer(uint8 CtrlIdx, uint8* BufIdxPtr, Eth_DataType** BufPtr, uint16* LenBytePtr)
- **Functions:**

i. Provides a transmit buffer resource. Granted buffer ID is stored to BufIdxPtr
ii. Address of the buffer is provided via BufPtr
iii. Length of the buffer required/allocated is LenBytePtr
iv. BUFREQ_E_BUSY return value indicates all buffers are occupied

5. Ethernet – Transmit API's - 4

- Std_ReturnType Eth_17_EthMac_Transmit(uint8 CtrlIdx, uint8 BufIdx,Eth_FrameType FrameType, boolean TxConfirmation, uint16 LenByte, const uint8* PhysAddrPtr)
- **Functions:**

i. Build the Ethernet Header (MAC Address and Frame Type)
ii. Payload is already updated by application.
iii. Updated the DMA Descriptor to transmit the frame.
iv. TxConfirmation indicates if confirmation of transmission is required.

6. Ethernet – Transmit API's - 3

- void Eth_17_EthMac_TxConfirmation(uint8 CtrlIdx)
- **Functions:**

i. Confirms all transmitted frames via callback API 'EthIf_Cbk_TxConfirmation'
ii. Free Tx buffers after application indication of confirmation
iii. To be called only in polling mode.
iv. EthIf_Cbk_TxConfirmation will be called from Tx ISR in interrupt mode.

7. Ethernet – Receive API's

- void Eth_17_EthMac_Receive(uint8 CtrlIdx)
- **Functions:**

i. Read the frames from all filled receive buffers
ii. Application call back 'EthIf_Cbk_RxIndication' for each frame received

iii. Multiple notification possible for one call of Eth_Receive

iv. To be called only in polling mode

v. Call back ‘EthIf_Cbk_RxIndication’ is called from Rx ISR

8. Ethernet – Status Reporting API’s - 1

- Std_ReturnType Eth_17_EthMac_GetControllerMode(uint8 CtrlIdx, Eth_ModeType* CtrlModePtr)
- **Functions:**

i. Reports the state of the MAC (MODE_DOWN / MODE_ACTIVE)

9. Ethernet – Status Reporting API’s - 4

- void Eth_17_EthMac_GetCounterState(uint8 CtrlIdx, uint16 CtrOffs, uint34* CtrValPtr)
- **Functions:**

i. Read the specified counter register (CtrOffs).

ii. Store the read value to CtrValPtr

10. Ethernet – Status Reporting API’s - 3

- void Eth_17_EthMac_GetCounterState(uint8 CtrlIdx, uint16 CtrOffs, uint34* CtrValPtr)
- **Functions:**

i. Read the specified counter register (CtrOffs).

ii. Store the read value to CtrValPtr

4.6 LwIP STACK OVERVIEW

LwIP (Light weight Internet Protocol) is an open source small and independent implementation of TCP/IP stack developed by Adam Dunkels at the Swedish Institute of Computer Science. This stack is freely available under BSD licence. It was designed for embedded systems as it focuses to reduce the resource usage while still having a full-scale TCP. And this makes it even more suitable for embedded systems with ten kilobytes of free RAM and room for around forty kilobytes of code ROM. This stack is written in C language and we can use it with or without an operating system. It rigorously reduces the processing and memory demands. It has various layers such as OS emulation layer, buffer and memory management, network interface functions. It has a packet buffer which is used an internal representation of a packet.

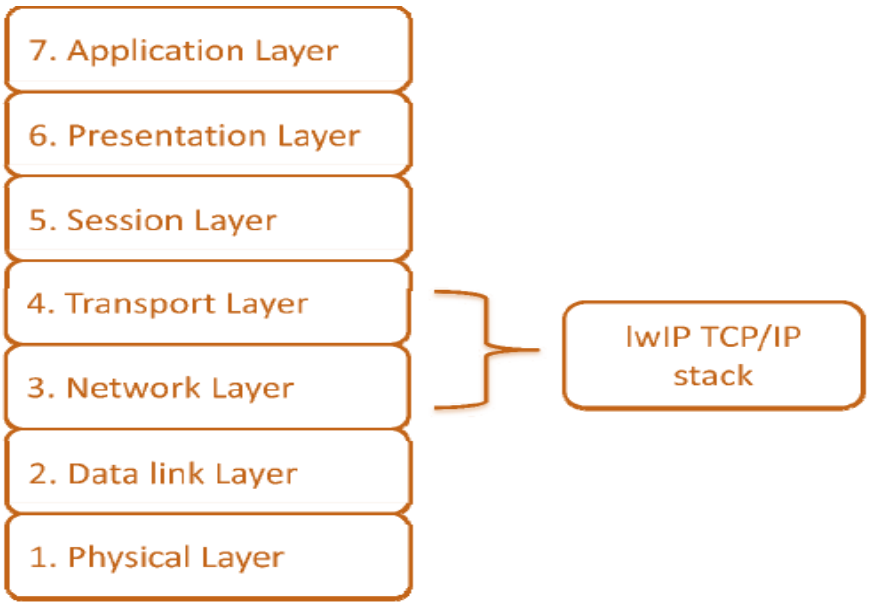


Fig 4.6: LwIP TCP/IP Stack Overview**.**

FEATURES:

The various features of LwIP stack are mentioned as follows:

- IP (Internet Protocol, IPv4 and IPv6) including packet forwarding over multiple network interfaces.
- ICMP (Internet Control Message Protocol) for network maintenance and debugging.
- IGMP (Internet Group Management Protocol) for multicast traffic management.

- MLD (Multicast listener discovery for IPv6) Aims to be compliant with RFC 2710.
- UDP (User Datagram Protocol) including experimental UDP-lite extensions.
- TCP (Transmission Control Protocol) with congestion control, RTT estimation and fast recovery/fast retransmit.
- Raw/ Native API for enhanced performance.
- Optional Berkeley- like socket API.
- DNS (Domain names resolver)

4.7 PROCESS MODEL

The process model of a protocol implementation describes the way in which manner the system has been divided into completely different processes. One method model that has been accustomed to implement communication protocols is to let every protocol run as a standalone method.

In the case of TC27x as soon as a receive packet request is initiated from the GMAC, the incoming request is handled by the Rx frame handler and is passed to the LwIP mailbox. The ethernet_init calls the mailbox_init (). The moment the mailbox receives the mail it initializes the LwIP stack. If the mailbox will be valid i.e. if its value holds 1 then the function will return a pointer to allocate the memory. The init_lwip () initializes the LwIP stack. The LwIP stack uses a call back function lwip_cbk_rxIndication () which uses a callback function to provide the receive buffer content in order to inform the LwIP that a buffer is filled with a frame. It will also calculate the eth_hdr, ip_hdr, and will also look for a VLAN frame. And based on the contents of the packet the request is handled by the respective individual application.

Similarly, during the transmission of a frame, the request is initiated a reverse manner as that of the frame reception. The application will raise a request in order to transmit the frame through the TX, which will initiate the LwIP to transmit the packet. LwIP will process the packet and will send the request to the TX frame handler and will send the packet stored in the buffer to the GMAC for further processing. The packet is transmitted using the Eth_transmit () API and it will also send a confirmation to the application using an Eth_txConfirmation () API.

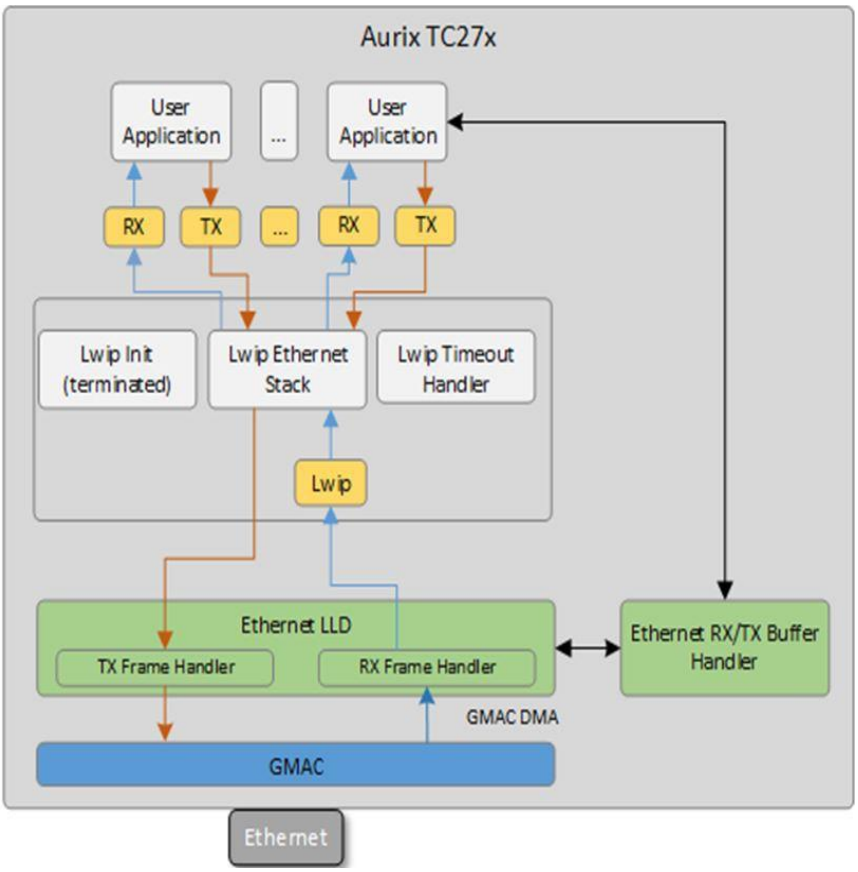


Fig 4.7: Ethernet flow with LwIP stack

4.8 PACKET BUFFER MANAGEMENT

The pbuf is a data structure used in LwIP which is essentially a linked list of buffers specially designed to represent one network packet. Using this data structure, the headers can be included without replicating the entire buffer. This data structure offers help for allocating dynamic memory to hold packet data, and for referencing data in static memory. A pbuf linked list is referred to as a pbuf chain. Packets are built from the pbuf data structure. It supports dynamic memory allocation for packet contents. Quick allocation for incoming packets is provided through pools with fixed sized pbufs. A packet may span over multiple pbufs, chained as a singly linked list. This is called a "pbuf chain". Multiple packets may be queued; also using this singly linked list. This is called a "packet queue". So, a packet queue comprises of one or extra pbuf chains, each of which incorporates at least one pbufs

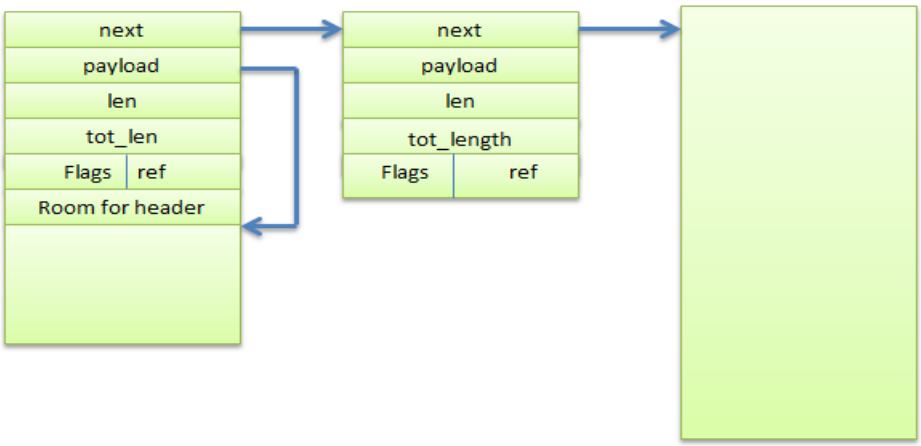


Fig 4.8: Pbuf structure

The pbuf structure has the following fields:

• Next: pointer to the next pbuf element in the pbuf linked list

• Payload: pointer to the data

• Len: length of the payload

• tot_len: sum of the length fields of the pbuf chain

• Flags: pbuf type

• Ref: reference count. A pbuf can only be freed if its reference count is zero

4.9 NETWORK CONNECTION API's

Various network connections API's have been used in order to make connection and some of them which have been used in the LwIP implementation are stated as follows:

1. Struct netconn * *netconn_new* (enum netconn_type t);

(i) Function: This function opens a netconn connection.

(ii) The In parameter used in the function is:

- t netconn_type where Netconn_type ={NETCONN_TCP, NETCONN_UDP, …}

(iii) The out parameter for the function is:

- Struct netconn

2. Struct netconn * netconn_new_with_callback (enum netconn_type t, netconn_callback callback);

(i) Function:

- This function creates a netconn connection handle.
- For receive or transmit events a callback is made available.

(ii) The In parameter used in the function is:

- t: netconn_type
- callback: function which is called when a frame is received

(iii) The out parameter for the function is:

- Struct netconn

3. Err_t netconn_bind (struct netconn * aNetConn, ip_addr_t * aAddr, u16_t aPort);

(i) Function:

- This function assigns to a generated connection handle it's IP address and Port number.

(ii) The In parameter used in the function is:

- aNetConn: the available connection handle
- aAddr: IP address of the connection of the local device itself
- aPort: Port number of the connection of the local device itself

(iii) The out parameter for the function is:

- err_t: ERR_OK if the assignment was achieved

4. Err_t netconn_send (struct netconn * aNetConn, struct netbuf * aNetBuf);

(i) Function:

- This function sends the data buffer in aNetBuf using the connection handle aNetConn.

- This send function can only be used for UDP protocols.

(ii) The In parameter used in the function is:

- aNetConn: the available connection handle
- aNetBuf: Pointer to the data buffer to be sent out

(iii) The out parameter for the function is:

- err_t: ERR_OK if the assignment was achieved

5. Err_t netconn_recv (struct netconn * aNetConn, struct netbuf ** aNetBuf);

(i) Function:

- Receives data in the buffer aNetBuf using the connection handle aNetConn.
- This send function can only be used for UDP protocols.
- The call of the function suspends the encapsulating task.
- When data is available the task wakes up.

(ii) The In parameter used in the function is:

- aNetConn: the available connection handle
- aNetBuf: Pointer to the data buffer to be sent out

(iii) The out parameter for the function is:

- err_t: ERR_OK if the assignment was achieved

6. Err_t netconn_delete (struct netconn * aNetConn);

(i) Function:

- This function will delete the connection once the data has been sent.

(ii) The In parameter used in the function is:

- aNetConn: the available connection handle

(iii) The out parameter for the function is:

- err_t: ERR_OK if the deletion was successful.

7. Err_t netconn_connect (struct netconn * aNetConn, ip_addr_t * aAddr, u16_t aPort);

(i) Function:

- This function attempts to establish a connection between the local device and the remote device.

(ii) The In parameter used in the function is:

- aNetConn: the available connection handle
- aAddr: IP address of the connection of the local device itself
- aPort: Port number of the connection of the local device itself

4.10 LwIP activities:

The LwIP task performs various activities during the course of its communication. It interfaces with application tasks and interrupt handler to perform various tasks. The following diagram gives a brief description about the LwIP task and how it works along with its sending and receiving activities respectively.

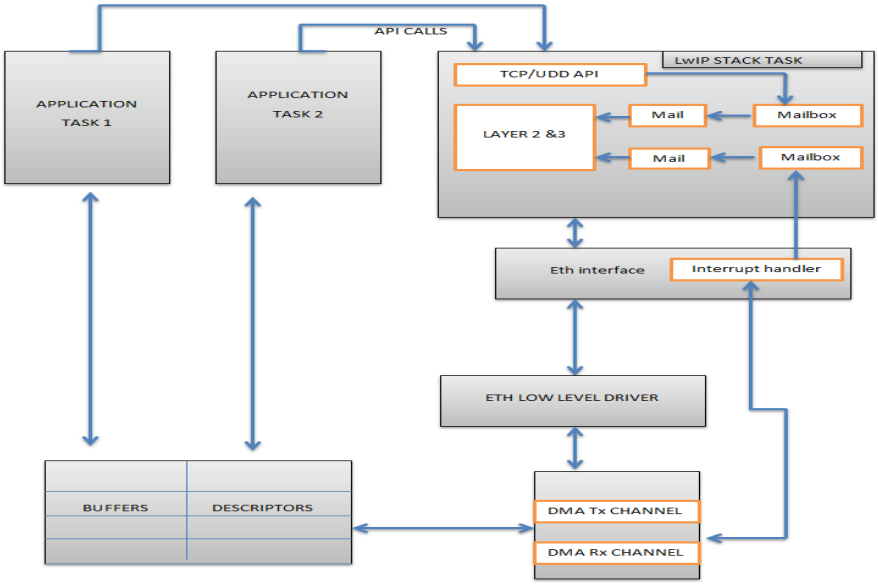


Fig 4.10: Stack function

The sequences of steps are mentioned as follows:

- Initially the LwIP task is woken up by a mail from the LwIP mail box.

- LwIP takes the pointer(s) to the packet from the mail.
- It then analyses the packet for the contents and applies various filters. Filter criteria are usually the IP address and the Port number of the packet.
- It decides on the next steps depending on the packet content.
- Layer 2 & 3 protocol requests (ARP, ICMP) are directly answered. Layer 4 protocol requests (TCP, UDP) are filtered & forwarded.
- If required by the protocol, an error answer frame may be initiated.
- The moment all the filters are passed then LwIP identifies the target task. And then LwIP sends a mail to the task mail box.
- The task mail box is created by the API call netconn_new (...).
- Meanwhile the application task has to releases the buffer(s) using an API call.
- And finally the LwIP triggers the Ethernet Interface to free the buffers.

The various receiving task activities are explained as follows:

- Receiving task launches LwIP API call err = netconn_recv (appl_netconn_ptr, &xRxNetBuf).
- This API call will deactivate the task.
- A UDP/TCP packet must be received and pass all filters in order to activate the task
- The LwIP puts the received UDP/TCP packet into the xRxNetBuf.
- And then finally the LwIP activates the task. The task will receive the UDP/TCP packet.

- And then the task deletes the received buffer (frees the memory).

The Ethernet Interface and Driver activities during this sequence of action are as follows:

- The Ethernet interrupt handles the DMA buffers
- Ethernet interrupt is triggered by the Rx DMA
- Ethernet interrupt uses the Ethernet Interface and LL driver
- Ethernet interrupt fetches the addresses of all filled buffers
- The buffers were filled by the DMA
- It clears the interrupt request(s)
- It requests a new mail resource from the LwIP mail box
- It puts the pointers of the frames into the mail resource
- It sends the filled box to the mail box handler
- It releases the emptied buffers to the DMA
- It sends a mail with the buffer addresses to the LwIP mail box
- The mail activates the LwIP task

The MAC Rx DMA stores the incoming bit stream

- DMA activates the descriptor available in the DMA
- DMA transfers the received buffer into the memory

- DMA sets the according status bits (incl. error bits)
- DMA fetches the next free descriptor
- If no descriptor is available the DMA stops receiving
- DMA triggers the DMA interrupt

4.10.1 PACKETS RECEIVE FLOW:

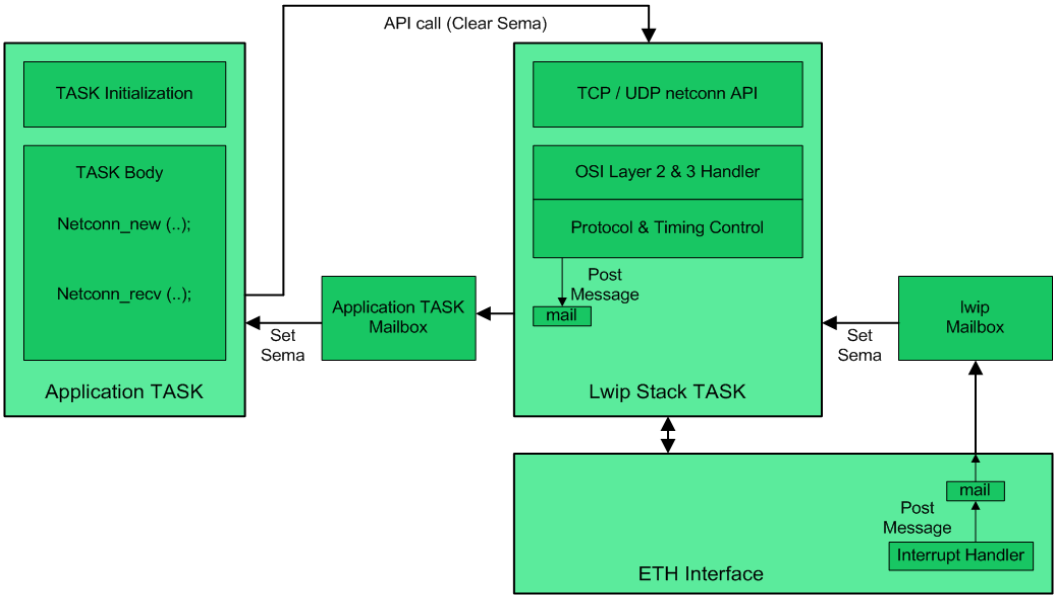


Fig 4.10.1: Packet receive flow

The sequence of API's used for the packet reception is mentioned as follows:

- appl_pc_netconn_RX_ptr = netconn_new(NETCONN_UDP);
- netconn_set_nonblocking(appl_pc_netconn_RX_ptr, 1);
- error = netconn_bind (appl_pc_netconn_RX_ptr, &appl_pc_local_ip_addr, locPcPort);
- error = netconn_recv(appl_pc_netconn_RX_ptr, &pPcRxNetBuf);
- netbuf_delete(pPcRxNetBuf

4.10.2 PACKETS SEND FLOW:

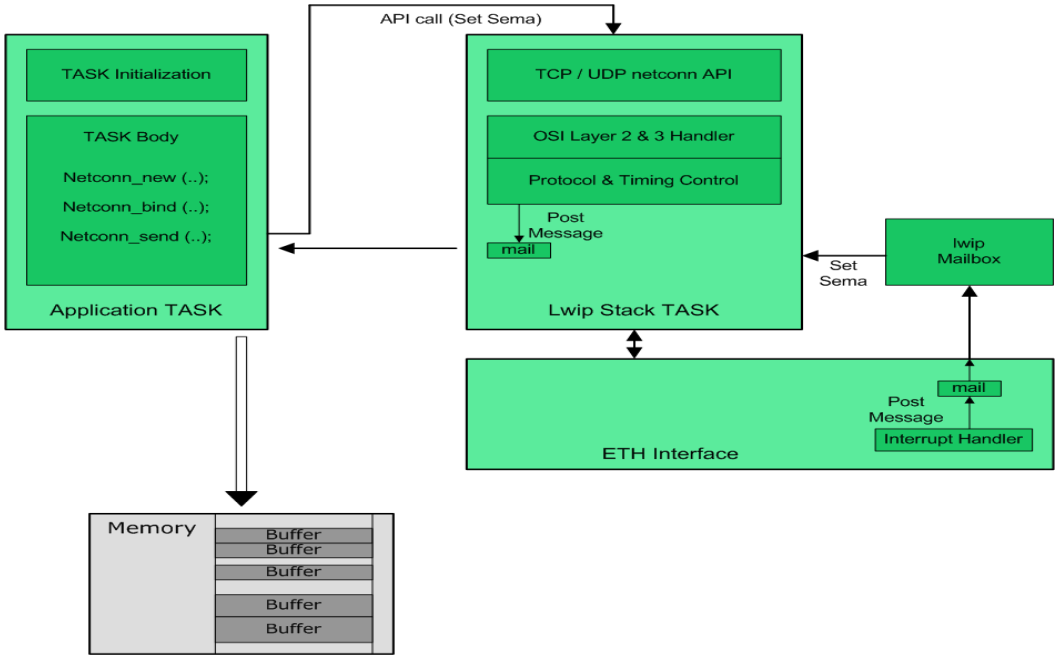


Fig 4.10.2: Packet transmit flow

The sequence of API's used for the packet transmission is mentioned as follows:

- appl_pc_netconn_TX_ptr = netconn_new (NETCONN_UDP);
- netconn_set_nonblocking(appl_pc_netconn_TX_ptr, 1);
- (error = netconn_bind (appl_pc_netconn_RX_ptr, &appl_pc_local_ip_addr, locPcPort)
- pPcTxNetBuf = netbuf_new();
- error = netbuf_ref(pPcTxNetBuf, 0, 0);
- pEthApplPcBuffer = ethApplPcBuffer + UDP_PACKET_TOTAL_LEN;
- error = netconn_connect(appl_pc_netconn_TX_ptr, &appl_pc_remote_ip_addr, remPcPort);
- error =netconn_send(appl_pc_netconn_TX_ptr, pPcTxNetBuf)

CHAPTER 5

IMPLEMENTATION OF VIRTUAL LAN

5.1 INTRODUCTION

VLAN is an IEEE 802.1q standard. It is a data link layer protocol. In simple terms, a VLAN could be a set of hosts within a concrete LAN that may interface with one another as though they were on a single, isolated LAN. By default, hosts in a specific VLAN cannot interface with hosts that are members of another VLAN, so for inter-VLAN communication, we need to have a router and likewise a valid VLAN ID so that the inter-VLAN communication can happen. With this configuration, each broadcast packet that has been transmitted is determined by each VLAN on the network which checks for its specific VLAN ID so as to forward that packet and work thereon otherwise it discards the packet once it fails to search the required VLAN ID.

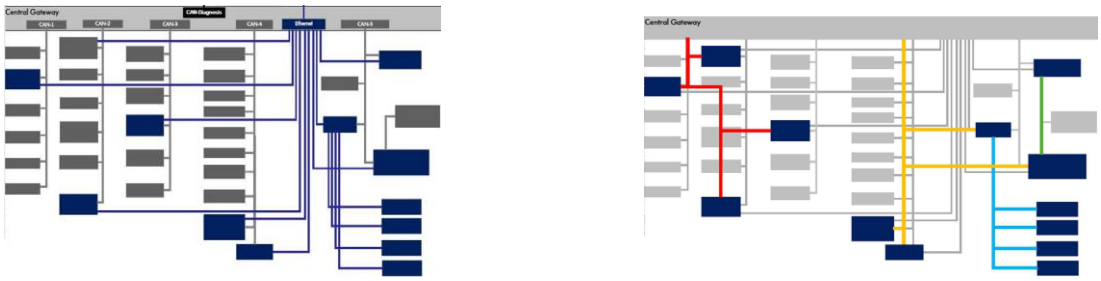


Fig 5.1: VLAN in automotive where the colored lines in the second figure represent different VLAN's

5.2 VLAN HEADER

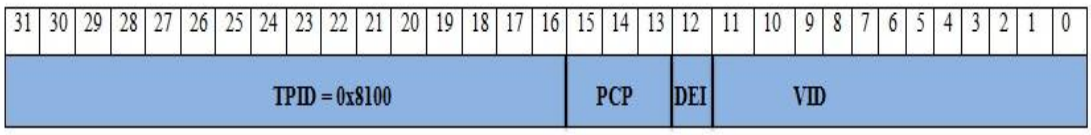


Fig 5.2.1: VLAN header

The figure shown above specifies the bit position for various fields in the VLAN tag. The VLAN tag is a 32-bit tag which can be majorly divided into two fields Tag Protocol Identifier and Tag Control Information. The Tag Protocol Identifier (TPID) is a 16-bit field that identifies the VLAN frame as an IEEE 802.1Q tagged frame. It is set to a value of 0x8100 which is the type field for a VLAN frame. Further, the Tag Control Information is also a 16-bit field and that is additionally divided into three sub fields named as Priority Code Point (PCP), Drop Eligible Indicator (DEI), and VLAN ID (VID). PCP is a 3-bit field that maps to the frame priority level. This value can be used to prioritize different classes of traffic in a network. The Priority allows optimization so that important information is forwarded preferentially. The DEI is a 1-bit field to indicate the frames which are eligible to be dropped in the presence of congestion. The DEI is typically additionally stated as Canonical Form indicator (CFI) Finally the VID is a 12-bit field specifying the VLAN to which the frame belongs. These three fields put together are categorized as the Tag control information.

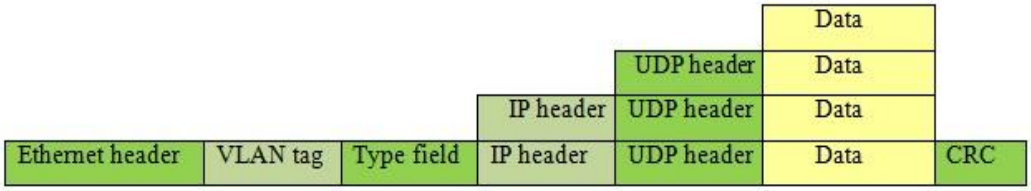


Fig 5.2.2: OSI stack structure

As we know that we've seven layers within the OSI model and each layer has its own frame format that is passed and processed from one layer to another as we move in an upward direction in the stack from physical layer towards the application layer. The ethernet header contains the MAC address of the packet, VLAN tag contains the information about VLAN ID and PCP, the type field contains the information about ether type, IP header contains the IPv6 or IPv6 header, UDP header contains the UDP or TCP header.

- At the Data link layer, we have an Ethernet frame as shown in the figure at the bottom level. The VLAN functions at the link layer so it is embedded in the Ethernet frame format at the link layer.
- As we move upwards towards the Network layer this ethernet frame format is

now referred to as IP frame which includes the information for IP header, UDP header, and the data.

- Further, as this IP frame is passed to the Transport layer where this frame is now referred to as UDP frame and we are left with only UDP header and data to be processed.
- And eventually, as we get into the application layer the data is processed.

5.3 VLAN TAGGING:

VLAN essentially permits us to have a view of multiple logical networks of a single physical ethernet (local area network). And therefore the foremost reason behind using this standard is to boost the network security by preventing the devices to access our LAN resources and secondly by doing this we can definitely improve the performance of our network. And this can be done by tagging the ethernet frames. VLAN tagging is a method of injecting VLAN ID into the header of a particular packet so that we can easily identify to which VLAN it belongs to. In fact tagging, the VLAN frame is nothing more than having a separate logical network as compared to a physical one. By doing this the traffic from one network is separated from the other network in a topology. By doing this we are limiting the broadcast traffic in a network and as a result providing a lot of security to the network.

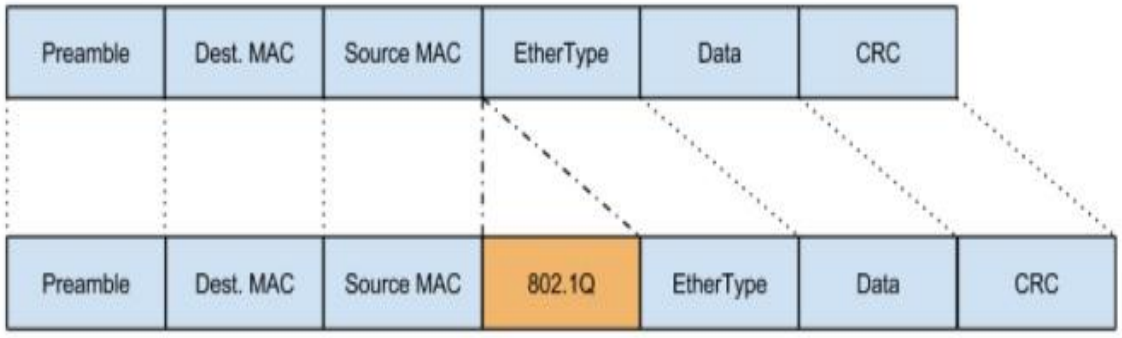


Fig 5.3: VLAN tagging in Ethernet header

VLAN enabled ports are typically categorized in one of two ways tagged or untagged. These can also be observed as "trunk" or "access" respectively. The purpose of a tagged or "trunked" port is to pass traffic for multiple VLAN's, whereas an untagged or "access"

port accepts traffic for only a single VLAN. Generally speaking, trunk ports will link switches, and access ports will link to end devices. An Ethernet interface can work as either an access port or a trunk port but one at a single time.

- An access port has only one VLAN configured on the interface; in other words, it carries traffic for only one VLAN. If this port receives a packet with a tag in the header; that port will drop the packet without even looking into the MAC source address.

- A trunk port can have two or more VLANs configured on the interface; it can carry traffic for several VLANs simultaneously In this case it will drop all the untagged traffic. This port is specifically enabled for VLAN tagging.

5.4 TYPES OF VLAN TRUNKING

1. **802.1Q Trunking**

IEEE 802.1Q is an open industry standard which is sometimes typically referred as 1Q or DOT1Q after the IEEE standard number is a frame tagging protocol. It embeds a 6-byte VLAN tag directly into the layer-2 frame header. This standard carries traffic for multiple VLAN's on a single interface between two Ethernet switches. This standard is for Ethernet networks only. 802.1q inserts a 6-byte field between the source MAC address and the Ether Type fields of the original Ethernet frame. Since a new field is added to the frame it also recalculates the CRC again for that frame. This tag will increase the size of Ethernet from 1516 bytes to 1518 bytes. This standard supports a maximum of 6096 VLAN's on a trunk port.

2. **ISL Trunking**

Inter Switched Link (ISL) is additionally used for an equivalent purpose as 802.1Q however it has a distinct frame format. It defines the point-to-point connection between the two devices. ISL encapsulation is performed with application specific integrated circuits (ASIC), therefore to support the ISL feature each connecting device has to be ISL configured. A non ISL device will not be able to consider an Ethernet frame carrying the ISL header. It also functions at Layer 2 of the OSI model like 802.1Q. It encapsulates the

entire Ethernet frame within the ISL header and trailer. ISL permits a maximum of 1000 VLANs on a trunk port. ISL encapsulate a frame with an additional header (26 bytes) and trailer (6 bytes). And as the ISL increases the frame size by 30 bytes, most of the switches will then discard the frame as being oversized.

5.5 VLAN MEMBERSHIP

1. Static VLAN

When we manually assign an individual or a group of ports to a VLAN, it's known as static assignment. By default, all the layer two switch ports are statically assigned to VLAN 1 until we alter the default configuration. Any host that is connected to that port will immediately become a member of that port. This is completely transparent to the host as it is unaware to which VLAN it belongs.

2. **Dynamic VLAN**

VLANs can also be allocated actively based on the MAC address of the host and this is known as dynamic assignment. This permits a host to remain in the same VLAN, regardless of which switch port it is connected to. Dynamic VLAN assignment needs a separate database to keep up the MAC-address-to-VLAN relationship. In additional refined systems, a user's network account can be used to verify VLAN membership, instead of a host's MAC address. Static VLAN assignment is far more common than dynamic VLAN.

5.6 VLAN WORKFLOW

We have two levels of filters the moment a packet has been received by the board. The first level of filter first looks for the VLAN tag in the header of the packet. And if a good VLAN tagged header is found only then the receiving task will forward the packet to the LwIP stack for further filtering of the packet. Consequently if the packet fails to have a valid VLAN tag attached in the header the packet is dropped their itself. Now as the packet passes the first level of filter then the LwIP stack further filters the packet based upon its content. It checks whether it is a PTP, AVB, VLAN, or a default frame and based on the content level 2 filters are applied by the LwIP stack and are passed to their respective queues to be processed further. Hence the packet has to pass these two levels of filters in order to be processed further. We additionally have another modification to

our implementation that if the receiving frame is not having a valid VLAN ID it will not be passed to the second level of filter irrespective of the fact that it is a valid Ethernet packet.

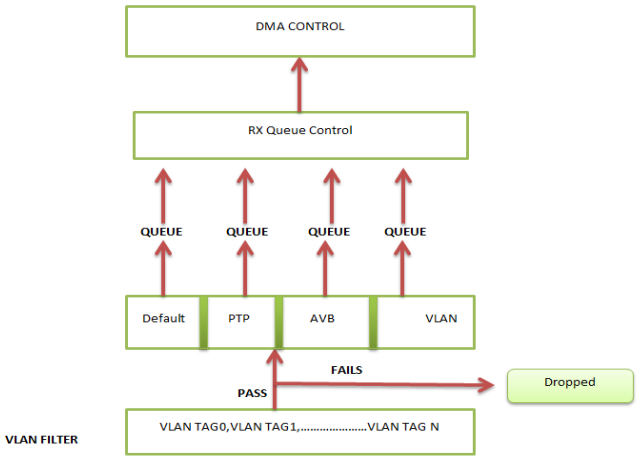
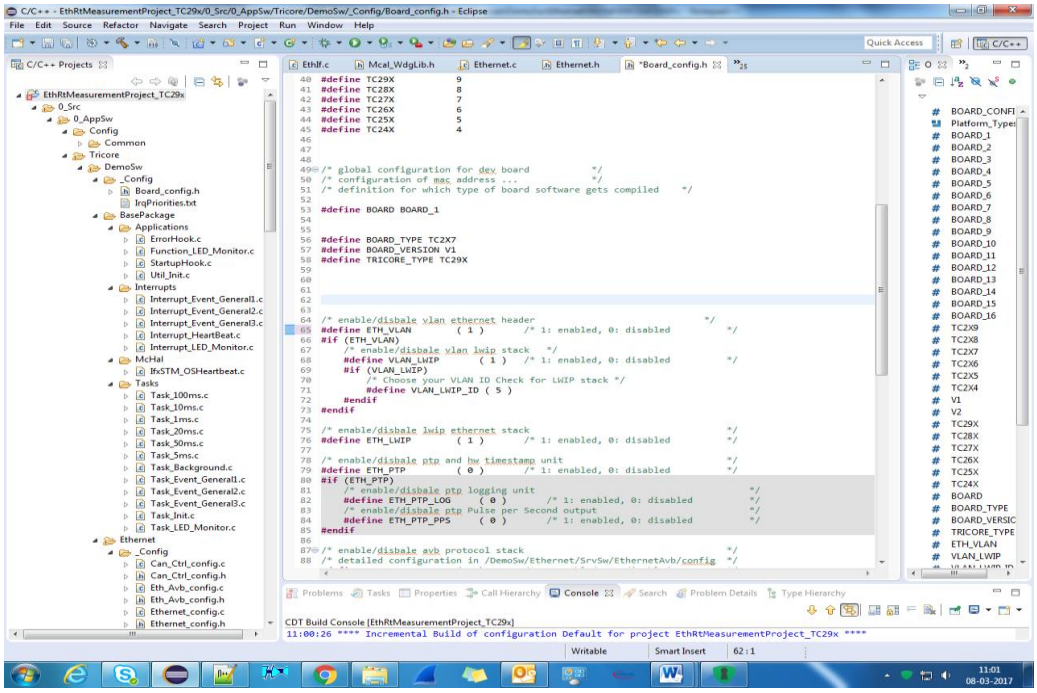
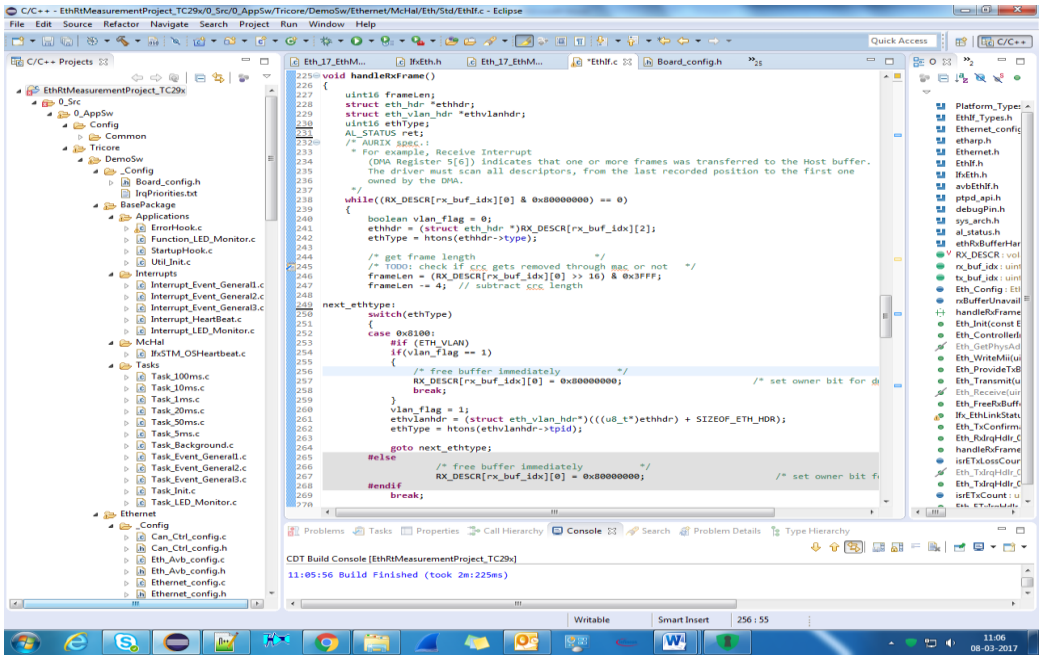


Fig 5.6: VLAN workflow

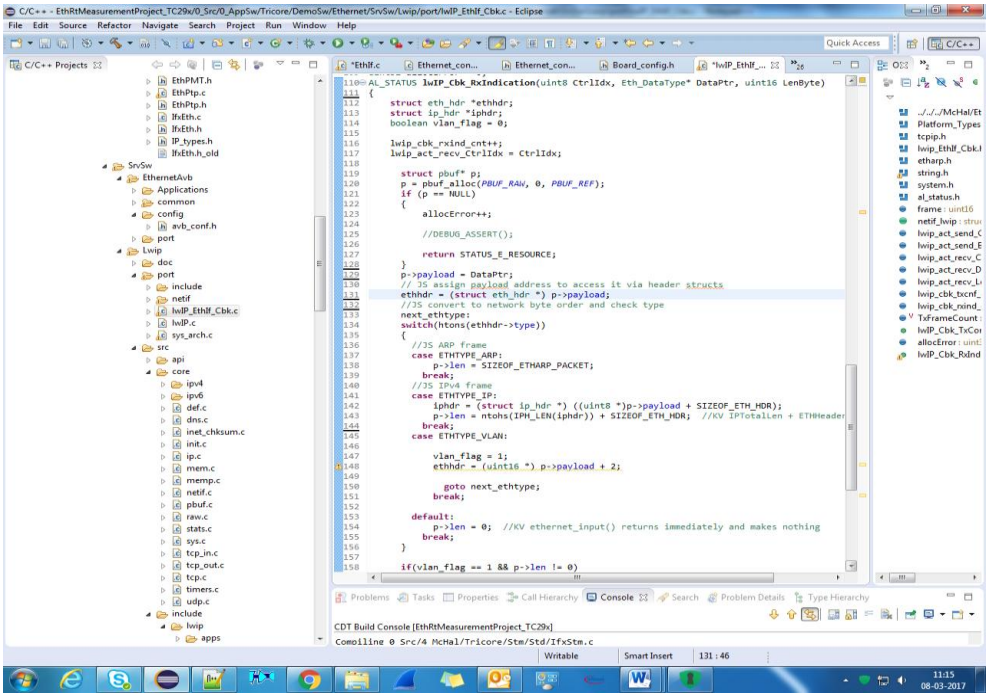
5.7 VLAN INTIALIZATION API'S



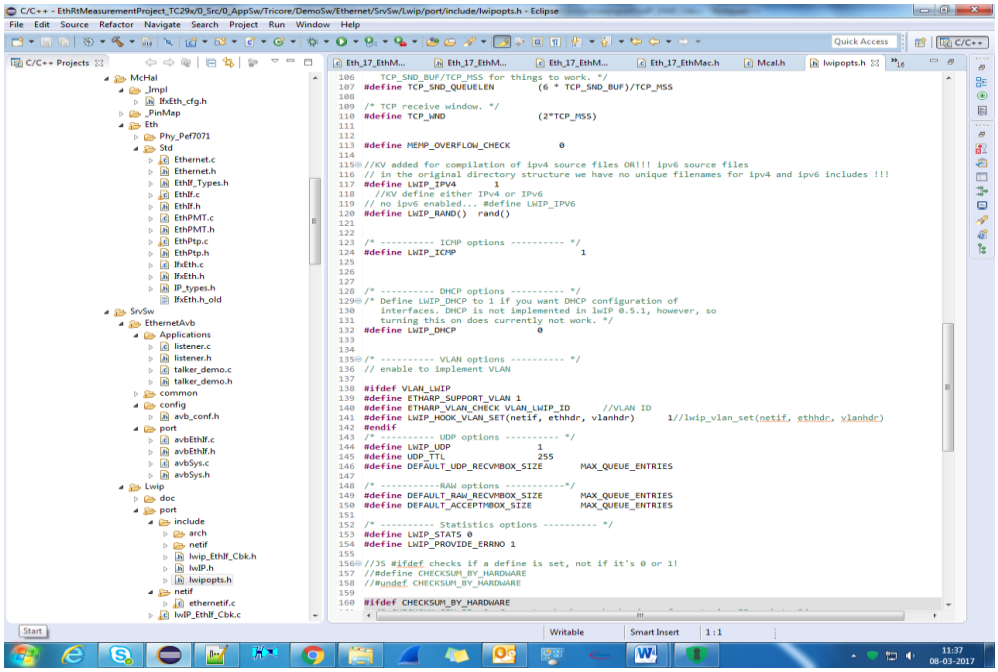
Screenshot 5.7.1: Section of code to enable and disable VLAN



Screenshot 5.7.2: Section of code where a VLAN frame is handled if ethType is a VLAN header



Screenshot 5.7.3: Call back function to provide the receiver buffer content. It informs the LwIP that a buffer is filled with a received frame



Screenshot 5.7.4: Section of code where two API's are being initialized

1. LWIP_HOOK_VLAN_CHECK (netif, eth_hdr, vlan_hdr) called from ethernet_input () if VLAN support is enabled.

Parameters:

- netif: struct netif on which the packet has been received
- eth_hdr: struct eth_hdr of the packet
- vlan_hdr: struct eth_vlan_hdr of the packet

Return values:

- 0: Packet must be dropped.
- != 0: Packet must be accepted

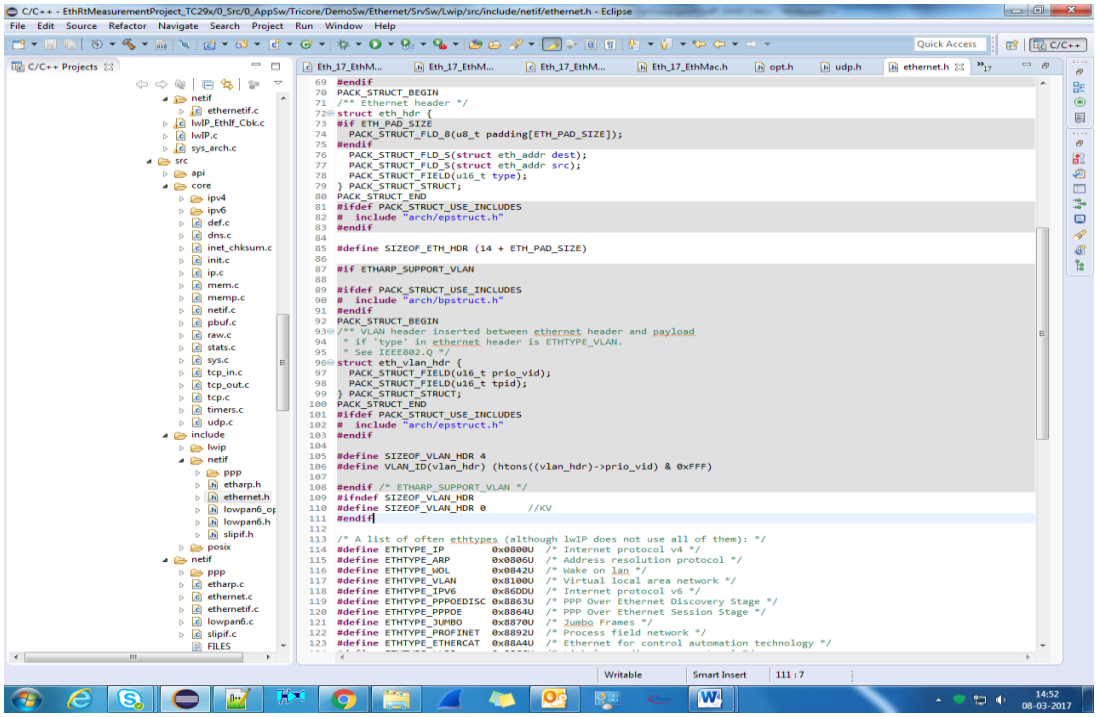
2. LWIP_HOOK_VLAN_SET (netif, eth_hdr, vlan_hdr) called from etharp_raw () and etharp_send_ip () if VLAN support is enabled.

Parameters:

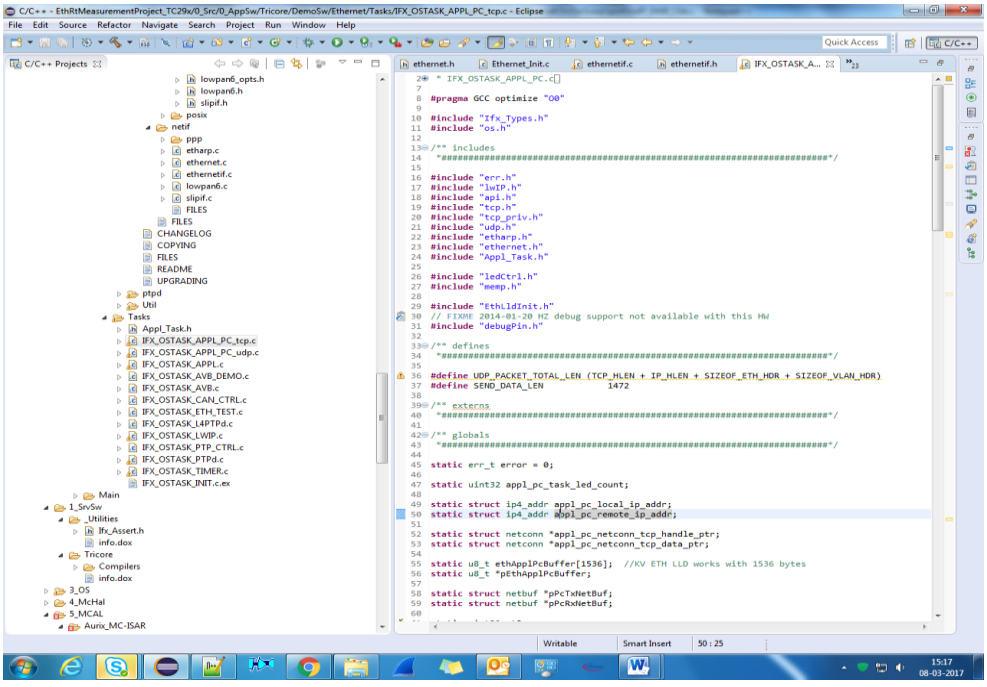
- Netif: struct netif that the packet will be sent through
- eth_hdr: struct eth_hdr of the packet
- vlan_hdr: struct eth_vlan_hdr of the packet

Return values:

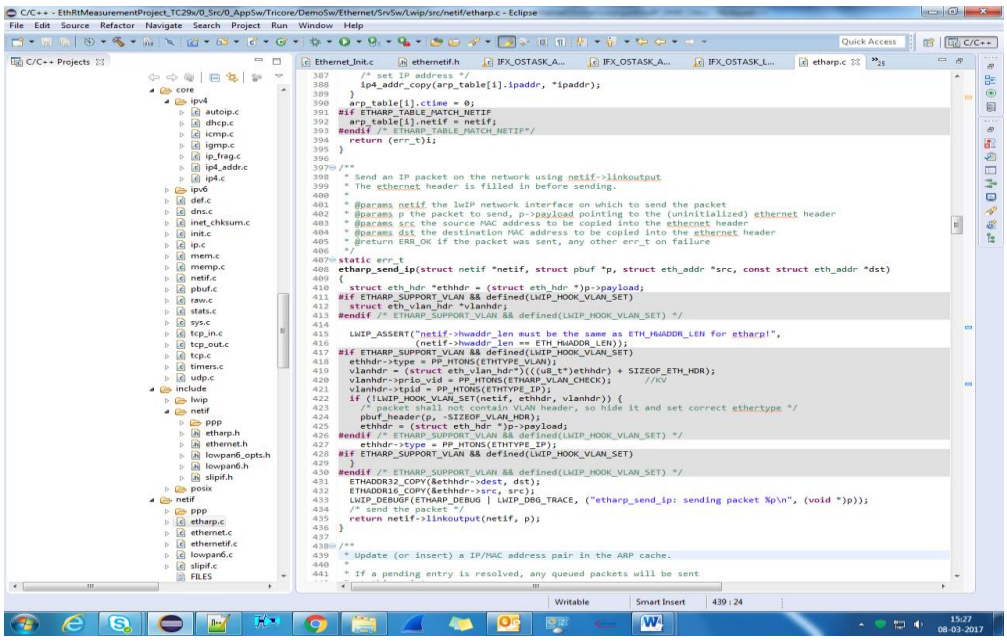
- 0: Packet shall not contain VLAN header.
- != 0: Packet shall contain VLAN header.
- Hook can be used to set prio_vid field of vlan_hdr.



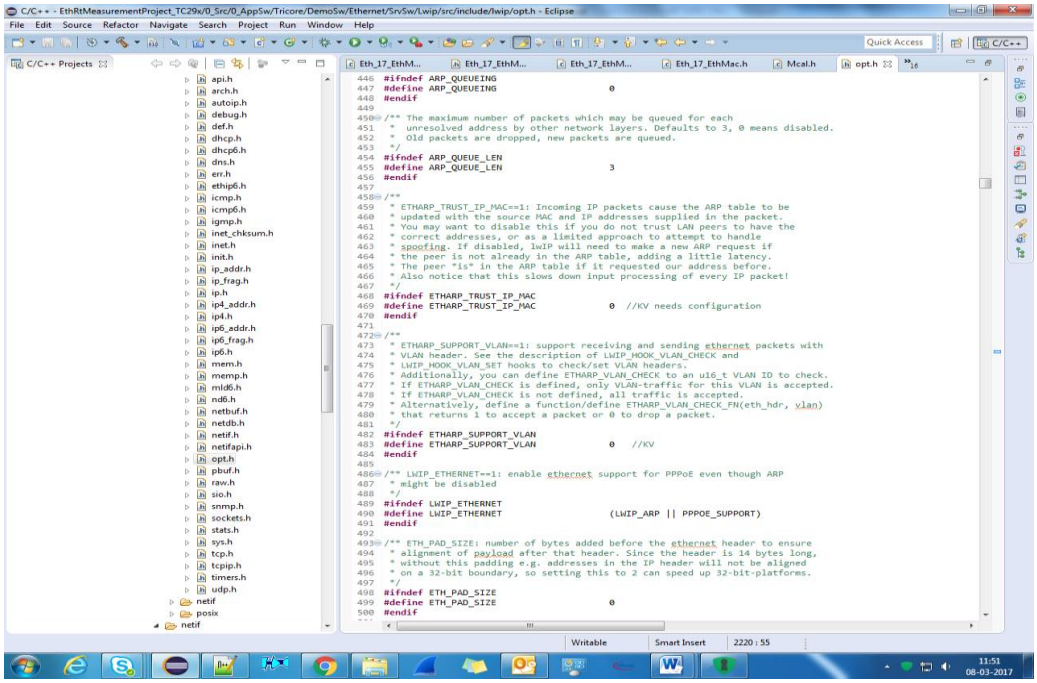
Screenshot 5.7.5: Section of code where a VLAN header and size of VLAN header is being defined.



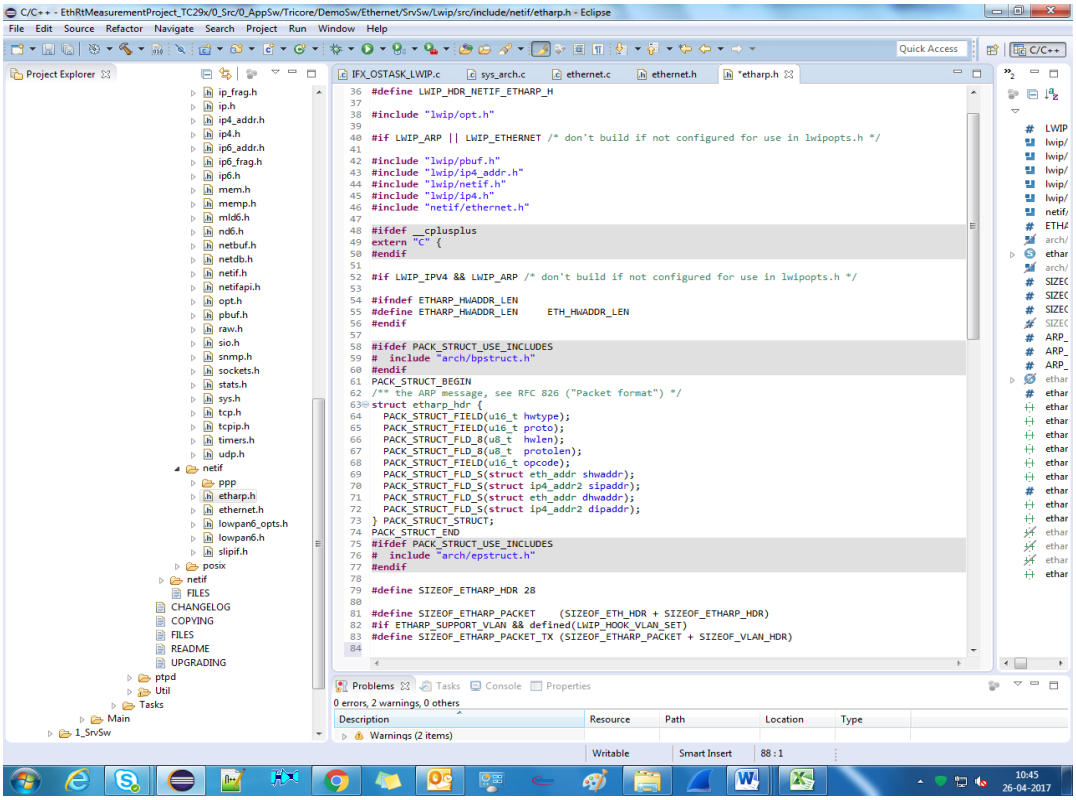
Screenshot 5.7.6: Section of code where the total length of the packet header is being increased as a VLAN header is being included in it.



Screenshot 5.7.7: Section of code where we send an IP packet on the network using netif->linkoutput. The ethernet header is filled in before sending.



Screenshot 5.7.8: Definition of ETHARP_SUPPORT_VLAN in opt.h file of LwIP



Screenshot 5.7.9: Section of code checking for a VLAN tag.

5.8 Implementation of VLAN

This implementation involves a sequence of steps which are required to be followed in order to compile and create an Elf file. Further we need to dump this Elf file on to the board in order to validate the implementation.

1. To make the program load and compile it with respective compiler and generate the required "ELF" file.

i. Start Eclipse

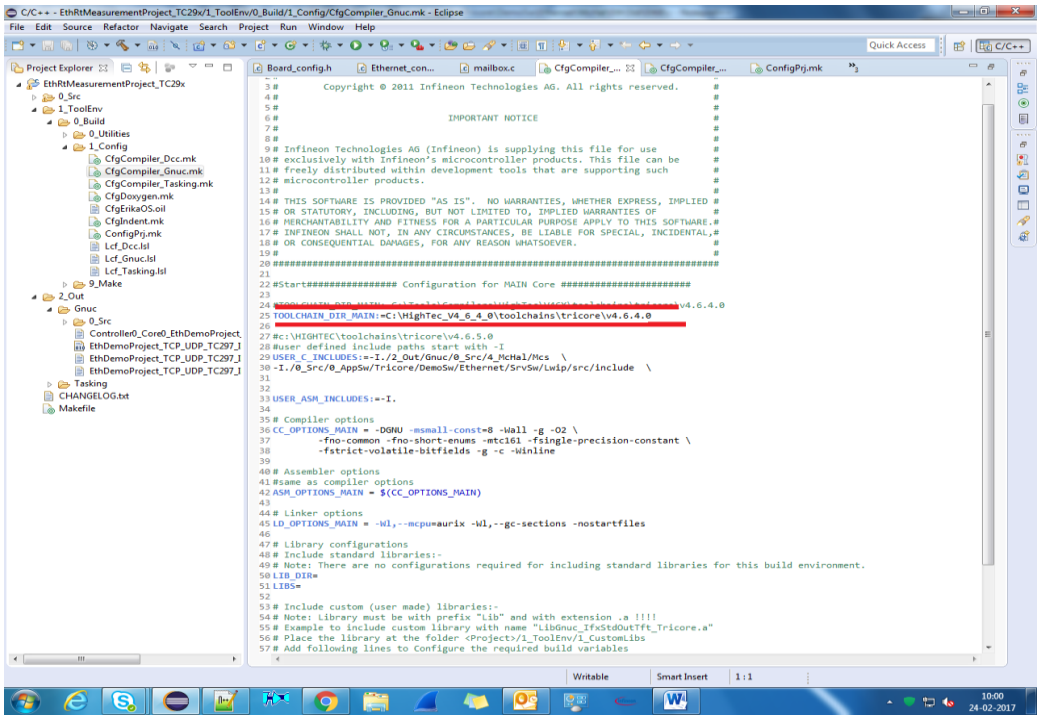
ii. Import the project from the corresponding folder by using the following steps:

File → Import → existing projects into workspace → Select root directory

↓

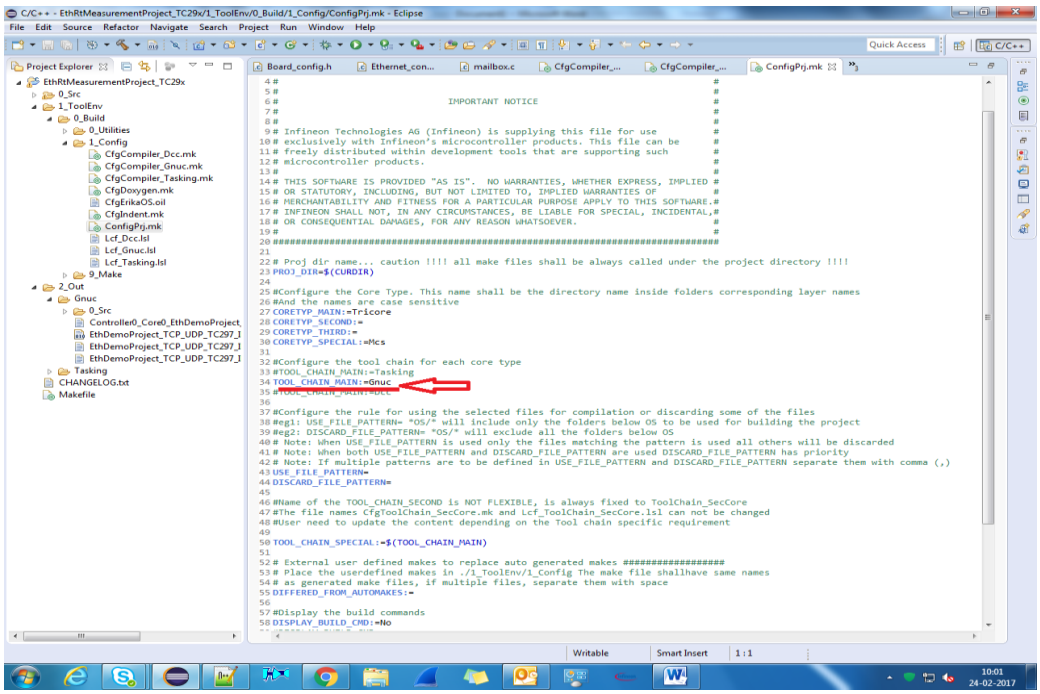
Click OK

iii. Expand the project and in the **1_ToolEnv** enable the respective compiler. I have used the GNUC compiler. Therefore in the **CfgCompiler_GNUC.mk** file enable the **"TOOLCHAIN_DIR_MAIN"** path for compiler.



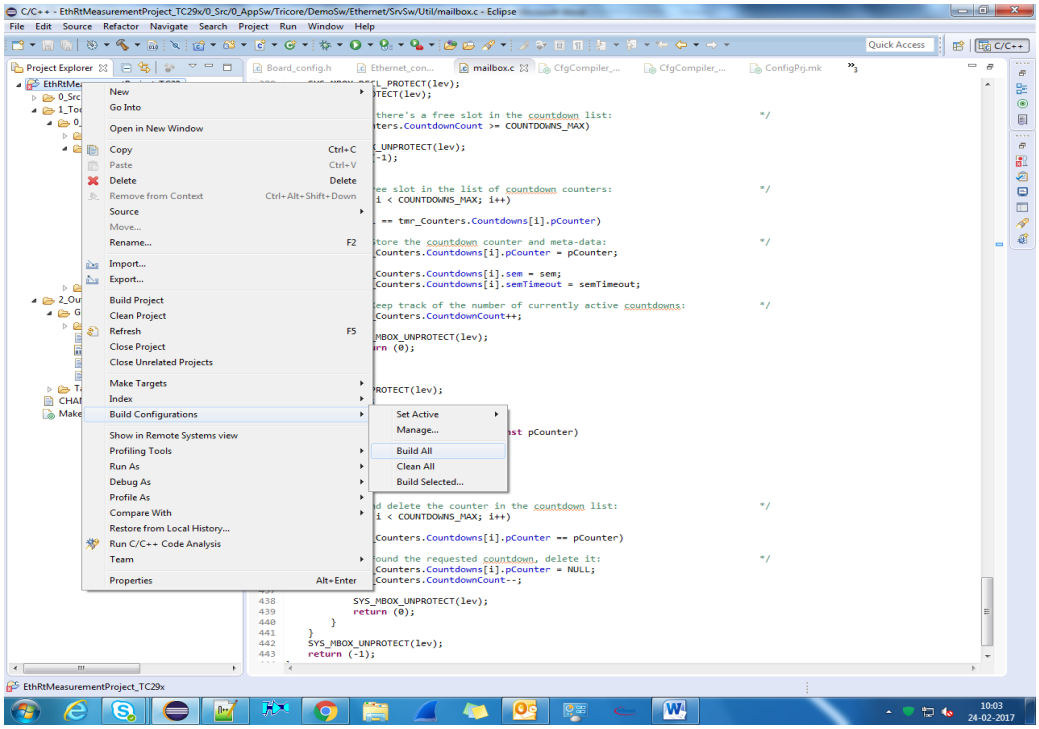
Screenshot 5.8.1: Screenshot to enable the required compiler for compiling

iv. Further enable the compiler in **ConfigPrj.mk** file as well.



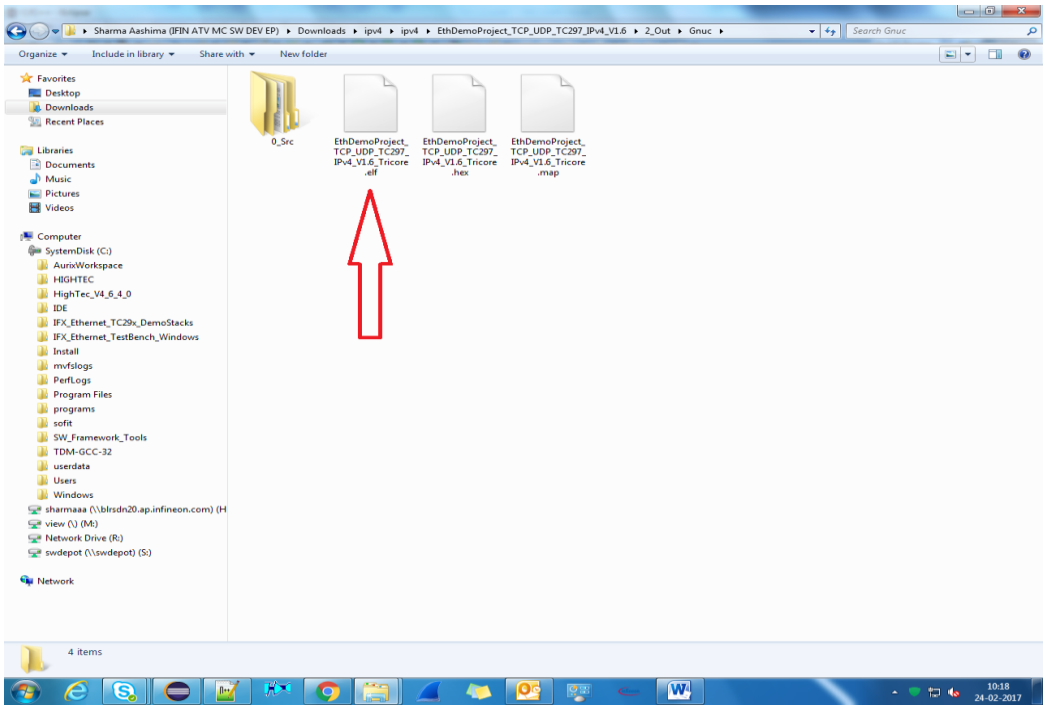
Screenshot 5.8.2: Screenshot to enable ConfigPrj file

v. Right click on the Project → Build Configurations → Build All.



Screenshot 5.8.3: Screenshot to build project

vi. After successful compilation, go to the home directory of the project, there in the **"2_OUT"** folder we can check the output files that are generated as a result of the compilation.



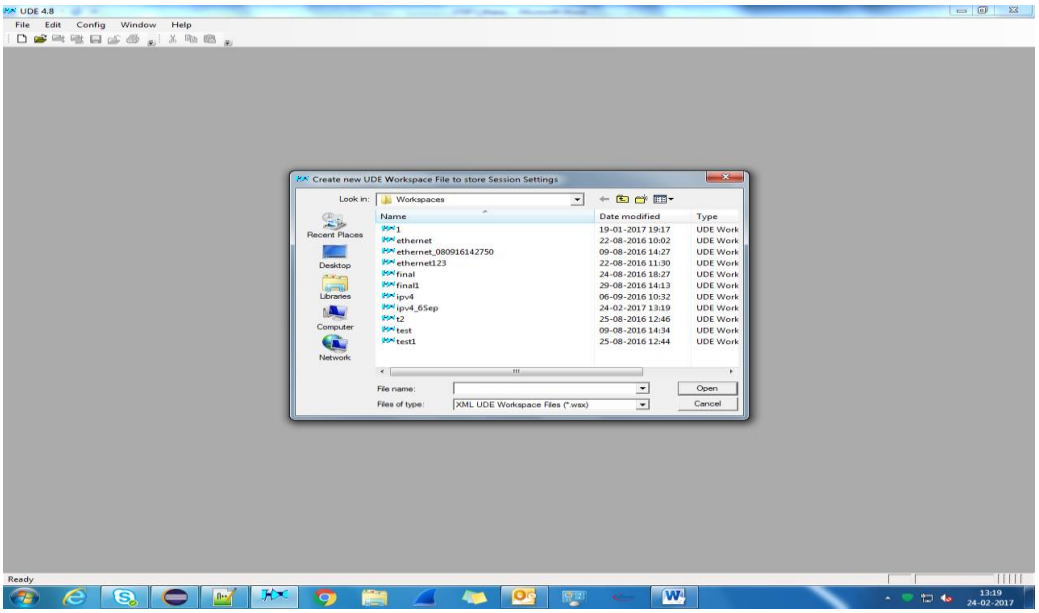
Screenshot 5.8.4: Generated output files after successful compilation

2. To flash the code on the testing board to further debug it.

i. Load UDE 4.8

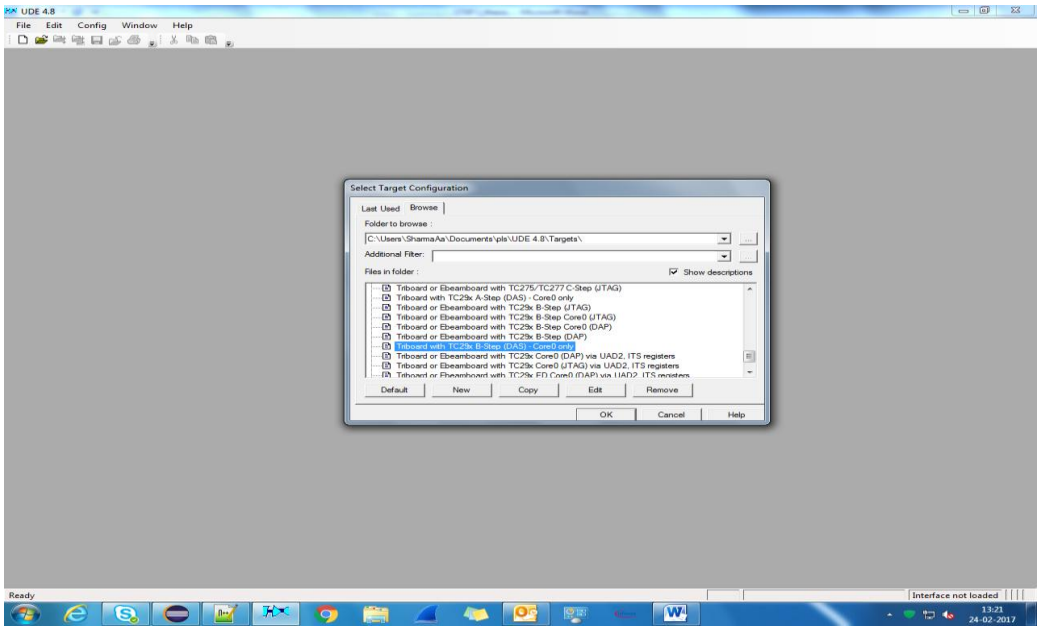
ii. Create a new workspace for the project by using the following steps:

File → New workspace → enter the name of the workspace → Click Open.



Screenshot 5.8.5: Creation of new workspace in UDE

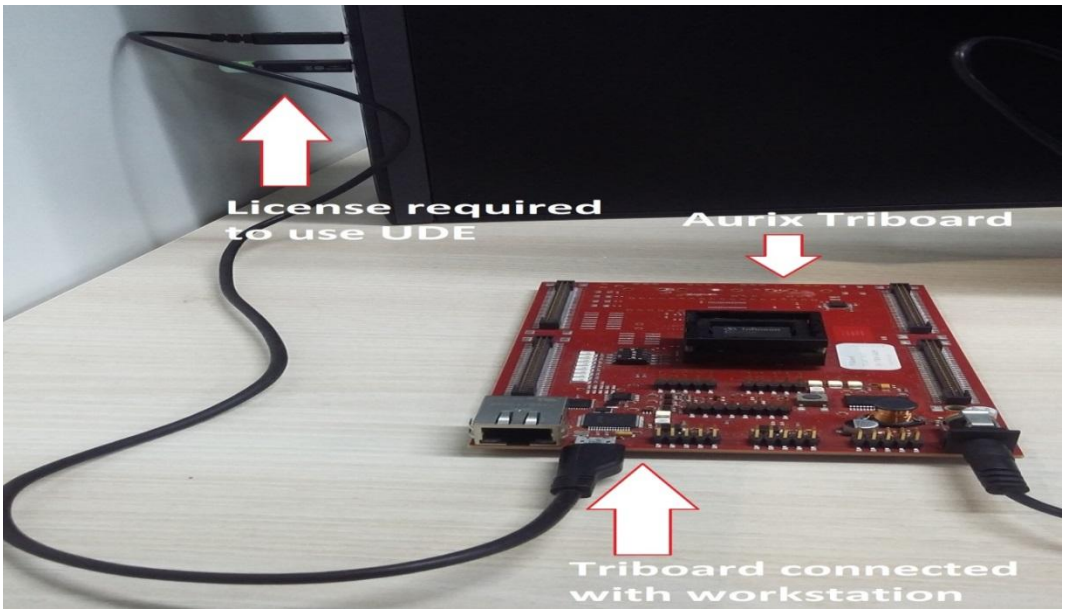
iii. Select the target configuration for the board and click OK.



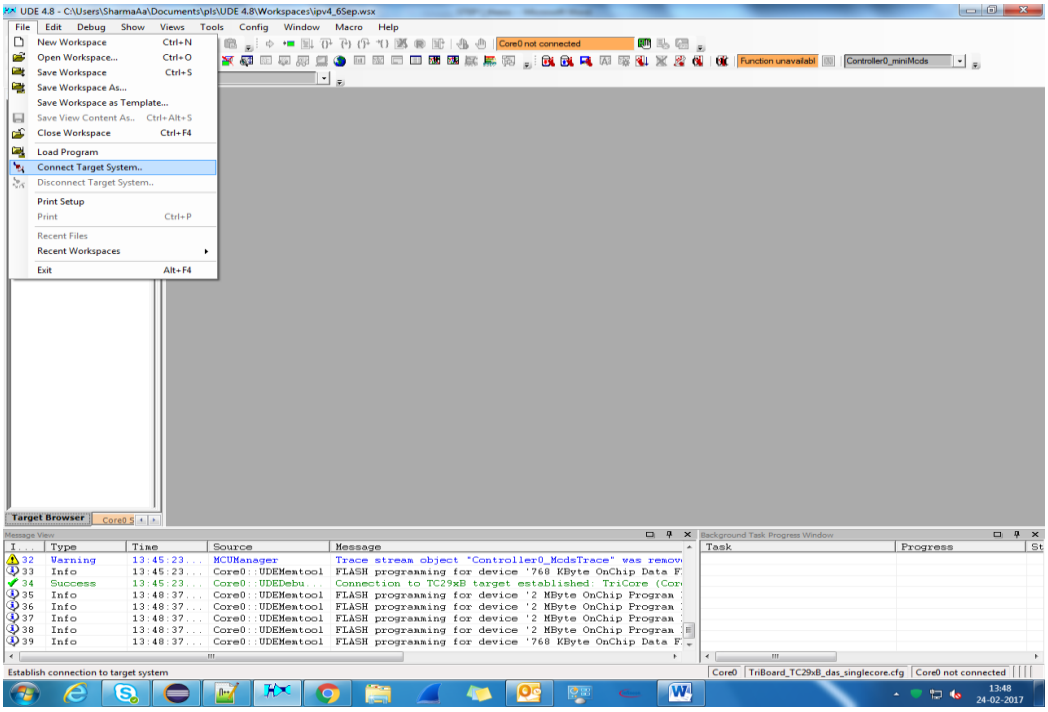
Screenshot 5.8.6: Selecting the target setup for loading the Elf file

iv. Next step is to connect the board with your workstation with a UDE license and the USB cable and adapter. After making the physical connection follow the steps:

File ⟶ Connect target system.



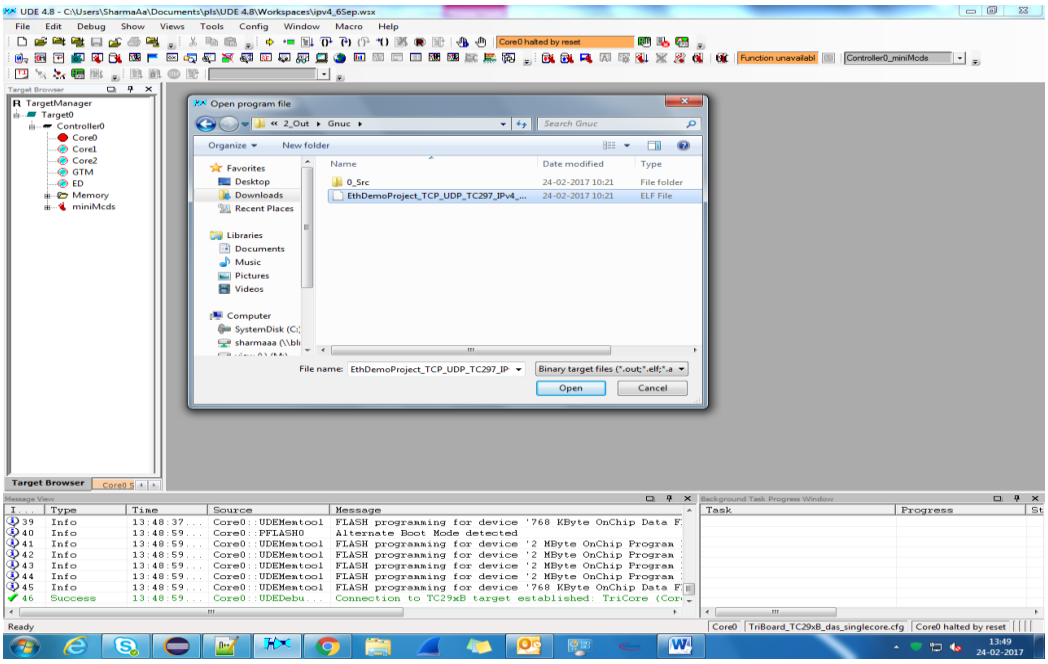
Screenshot 5.8.7: Setting up the hardware for testing



Screenshot 5.8.8: Connecting the target system with the UDE

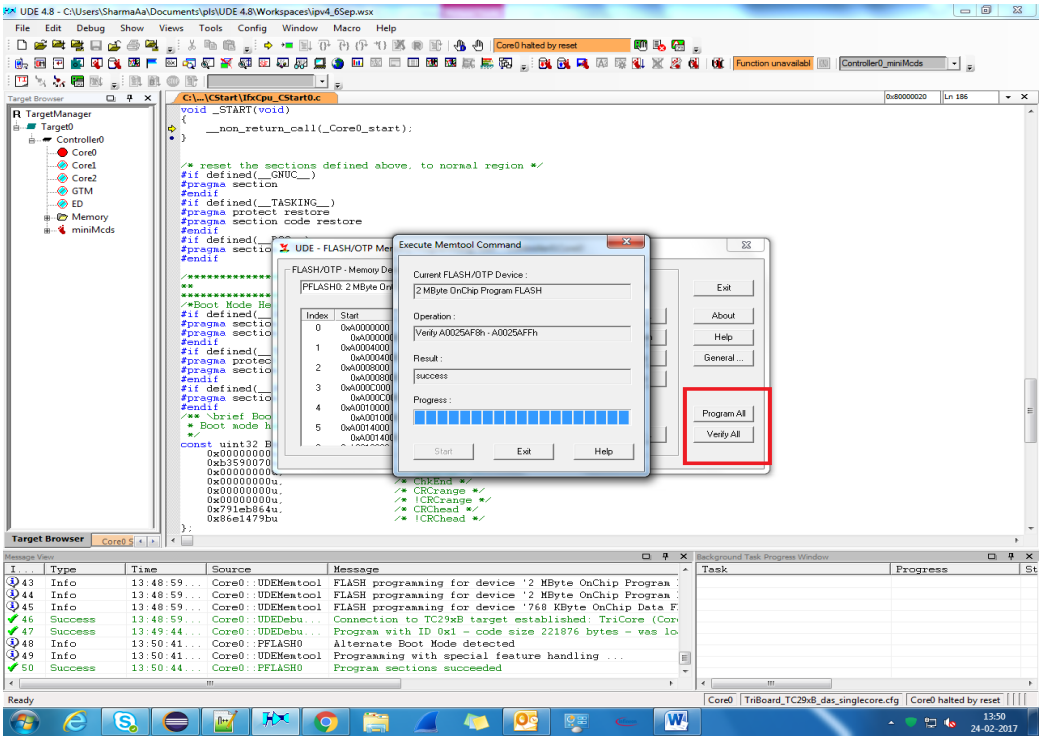
v. After the successful connection load the **"ELF"** file. To do the same follow the steps:

vi. File ⟶ Load program ⟶ Select the file from the directory ⟶ Click OK.



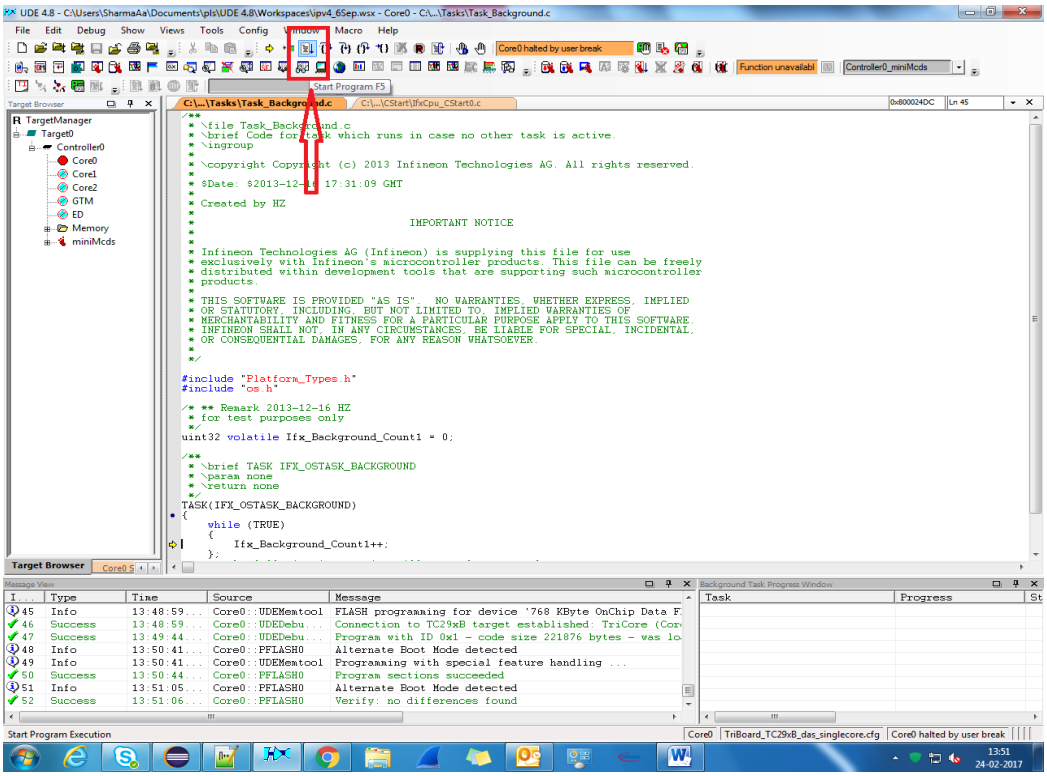
Screenshot 5.8.9: Loading the Elf file into the board

vii. After loading the file in the UDE, flash it to the board using **Program All.**

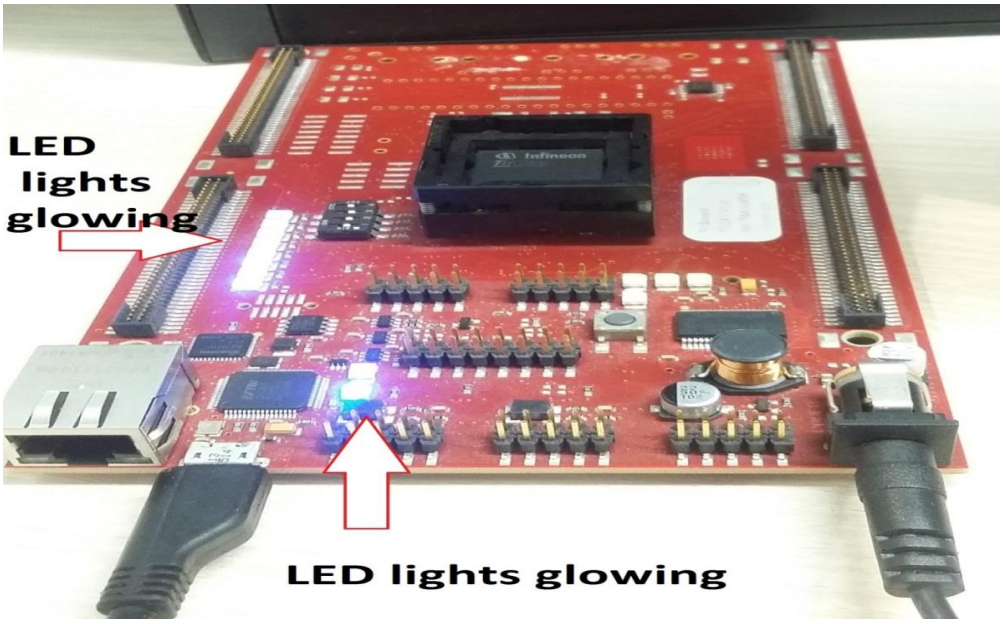


Screenshot 5.8.10: Final dumping of code onto the board

viii. Click on **Start Program** for final execution.



Screenshot 5.8.11: Screenshot representing how to start the final execution



Screenshot 5.8.12: LED's glowing after successful loading the ELF file onto the board

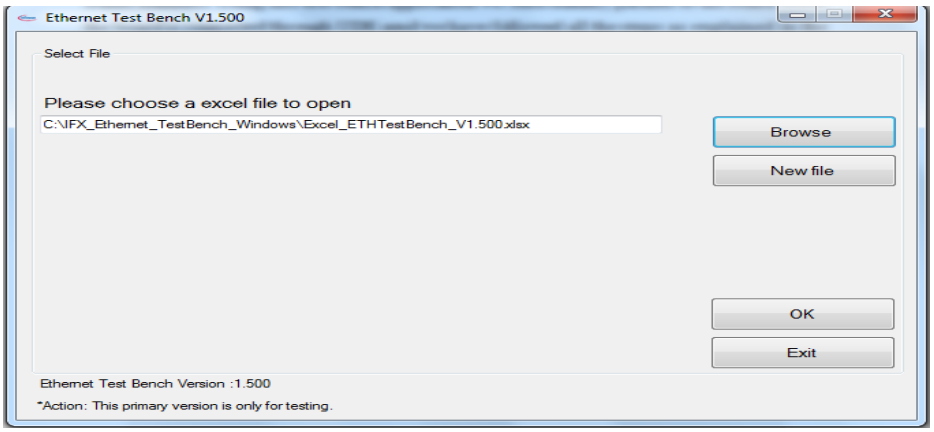
CHAPTER 6

EXPERIMENTAL OBSERVATIONS AND DISCUSSIONS

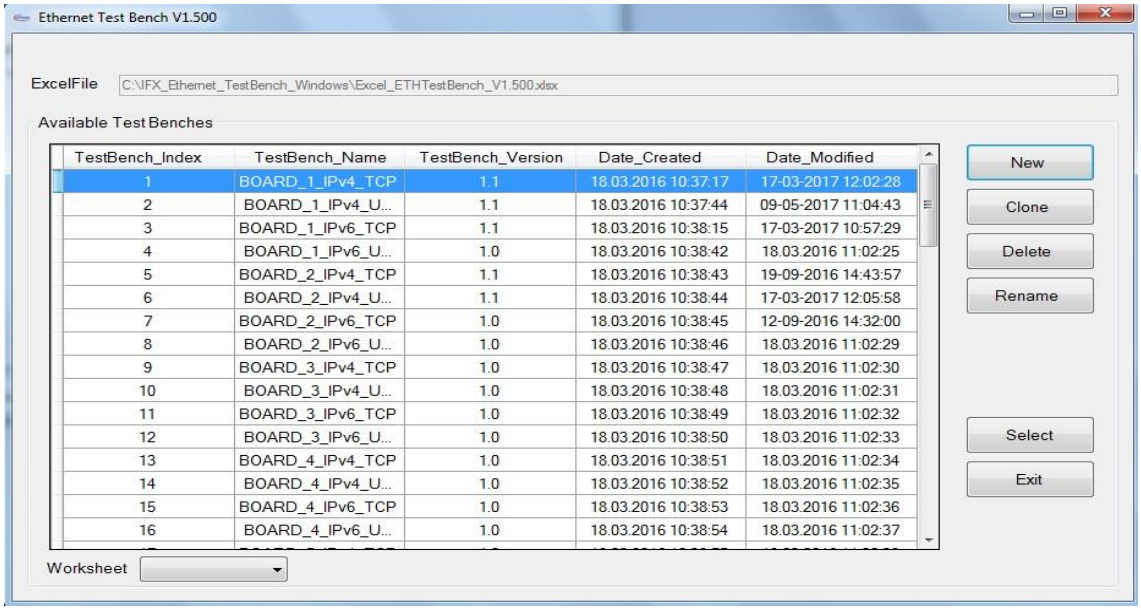
As the code has been implemented on Aurix hardware with the prevailing MCAL setup, Infineon's Test Bench (Internal Test Simulation environment) is used to test the code for the desired results. This Test bench is a windows based application in which an excel file is imported which is essentially a file that contains configuration parameters for the various boards and also the desktop. For example it already contains information about the MAC address and IP address of the board and the PC, port numbers, transmission frame size and transmission frame numbers. Using this test bench application dummy packets are sent to the board while the board is connected through UDE and we have followed all the steps as explained within the implementation section. Once ready with the desired setup dummy frames are sent to the board and eventually the results are traced through the network analyzer tool "WIRESHARK". Therefore the various steps which have been followed to test the setup are explained as follows:

1. TO STEP UP THE TEST ENVIRONMENT

(i) Firstly the excel file is imported into the test bench application that sets the assorted parameters required to send the packet to the board from the test bench.

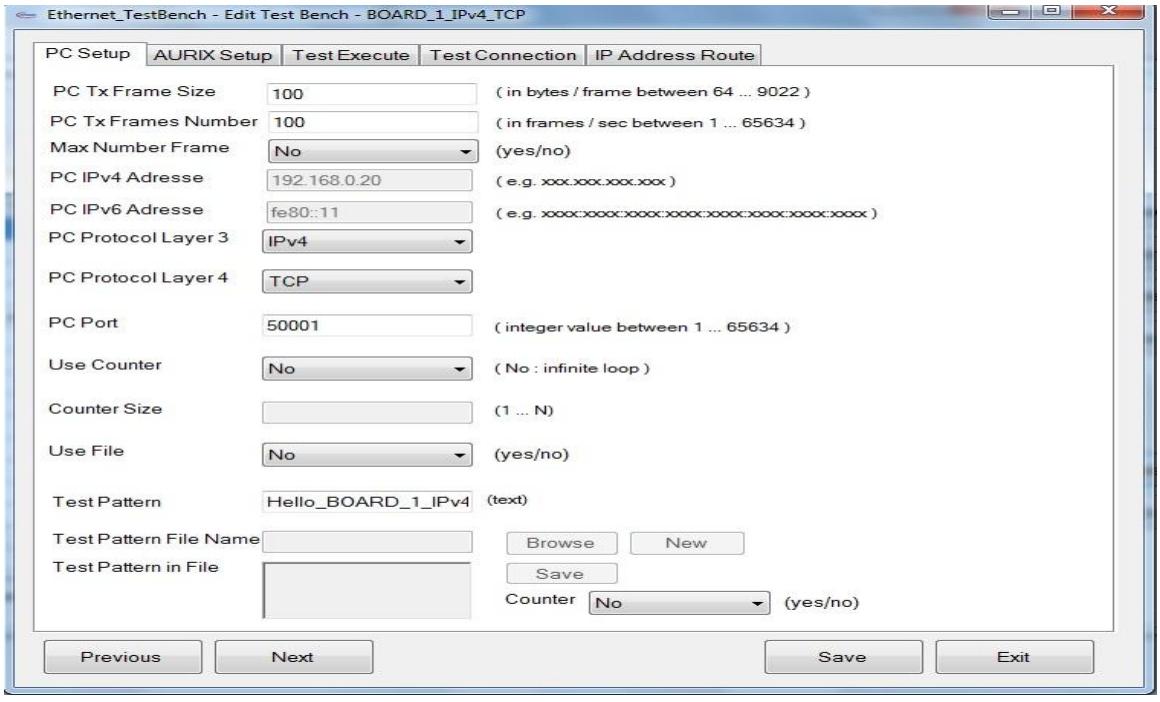


Screenshot 6.1: Importing excel file in the test bench



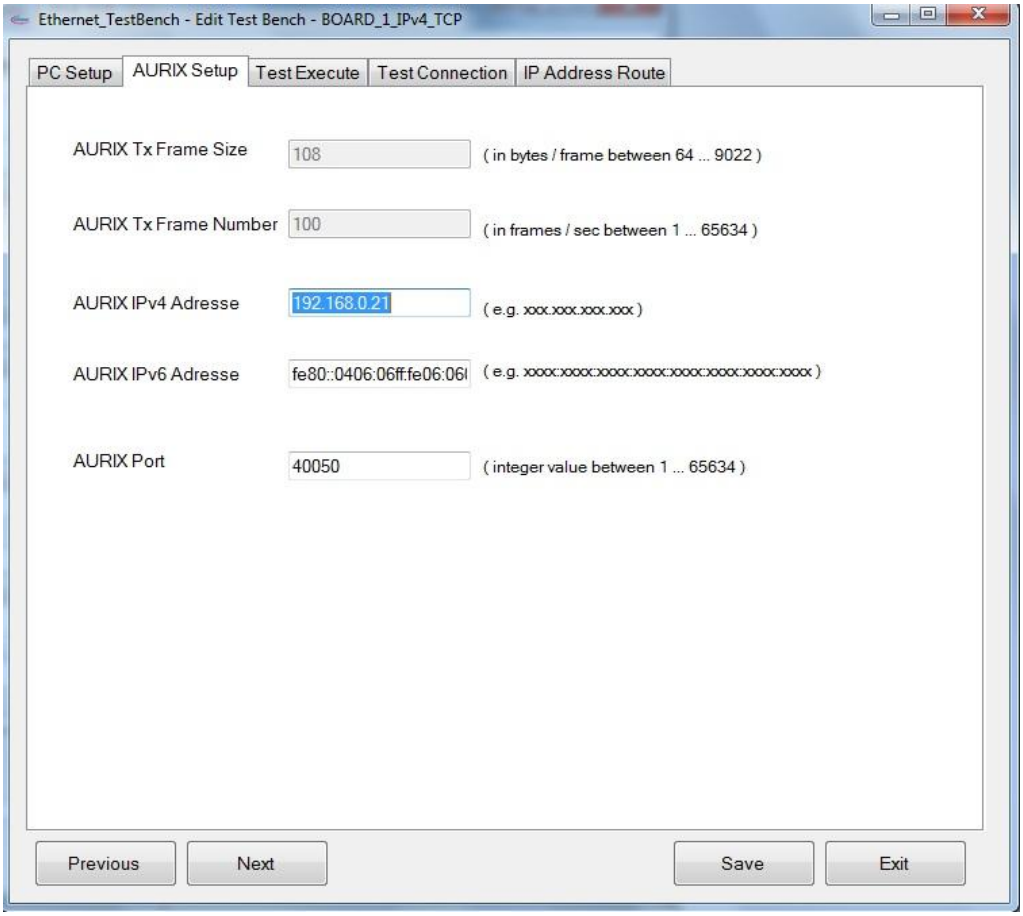
Screenshot 6.2: List of various boards configured in the excel file for selection in the test bench

(ii) Once the excel file has been imported some of the parameters can be changed for the PC manually as well in the PC Setup option. It offers the details about the kind of protocol being tested, IPv4 address, port number, test patterns and varied alternative parameters as shown.



Screenshot 6.3: PC setup tab

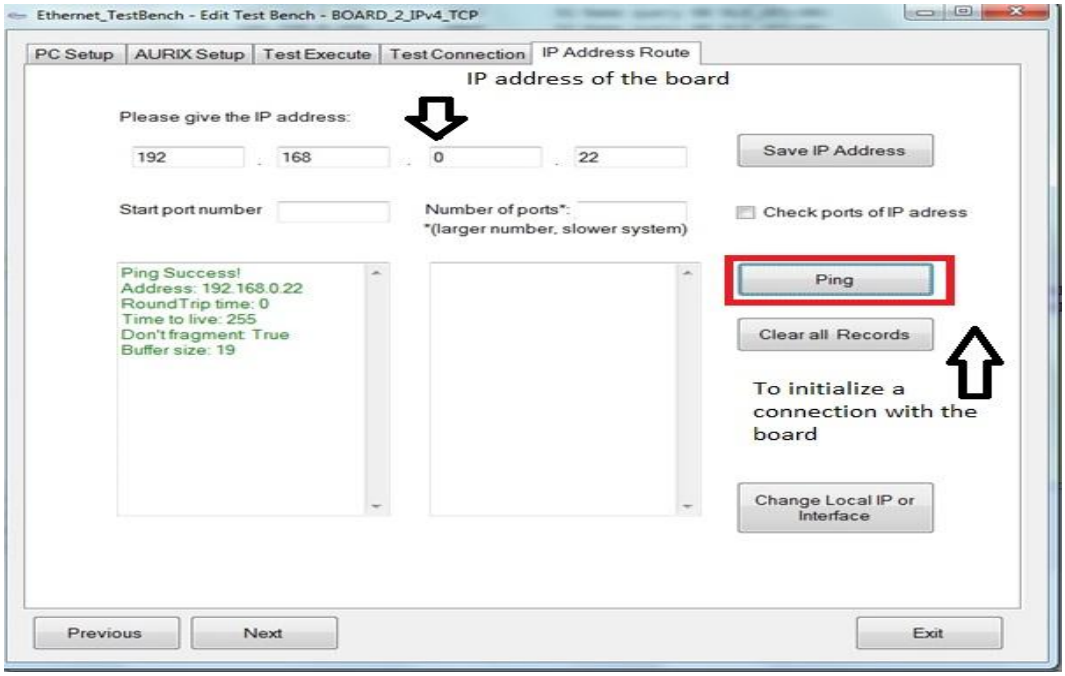
(iii) As the parameters can be modified for the PC in a similar fashion we also have the tendency to even alter the parameters if required for the Aurix board that is connected with the UDE for the testing purpose. It also includes numerous parameters within the AURIX setup tab like Aurix IPv4 address, port number, frame size etc.



Screenshot 6.4: AURIX setup tab

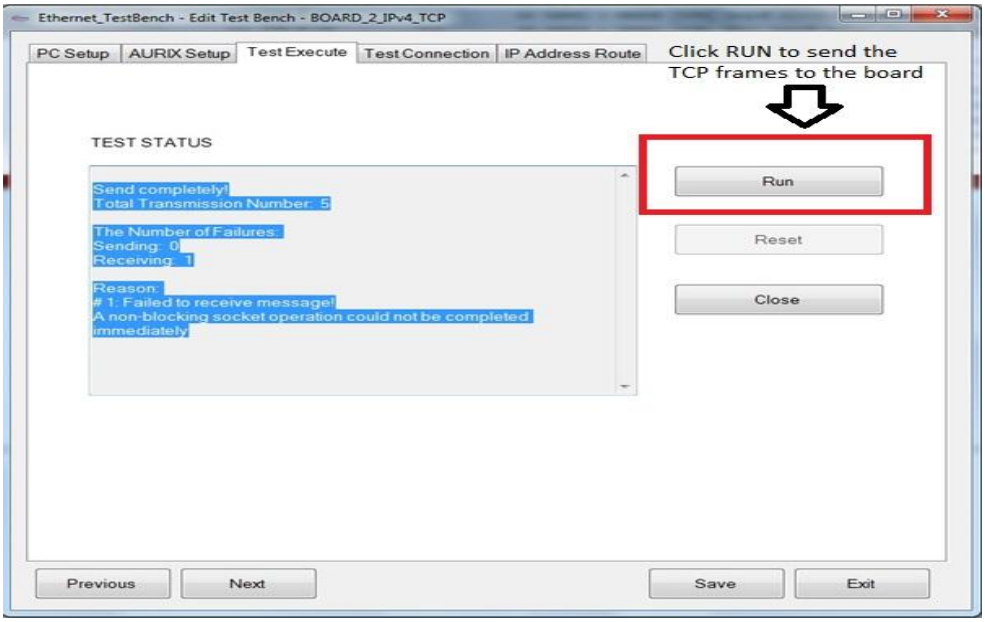
(iv) Once ready with the parameters now we can test the ARP and ICMP packets using IP address route option. Here we fill in the IP address of the board and once the IP address of the board is saved then click on the “Ping” option, the test bench sends ARP request packet to the board looking for the connection

to be built. And if it is successful a confirmation is received about the connection in the test box alongside.



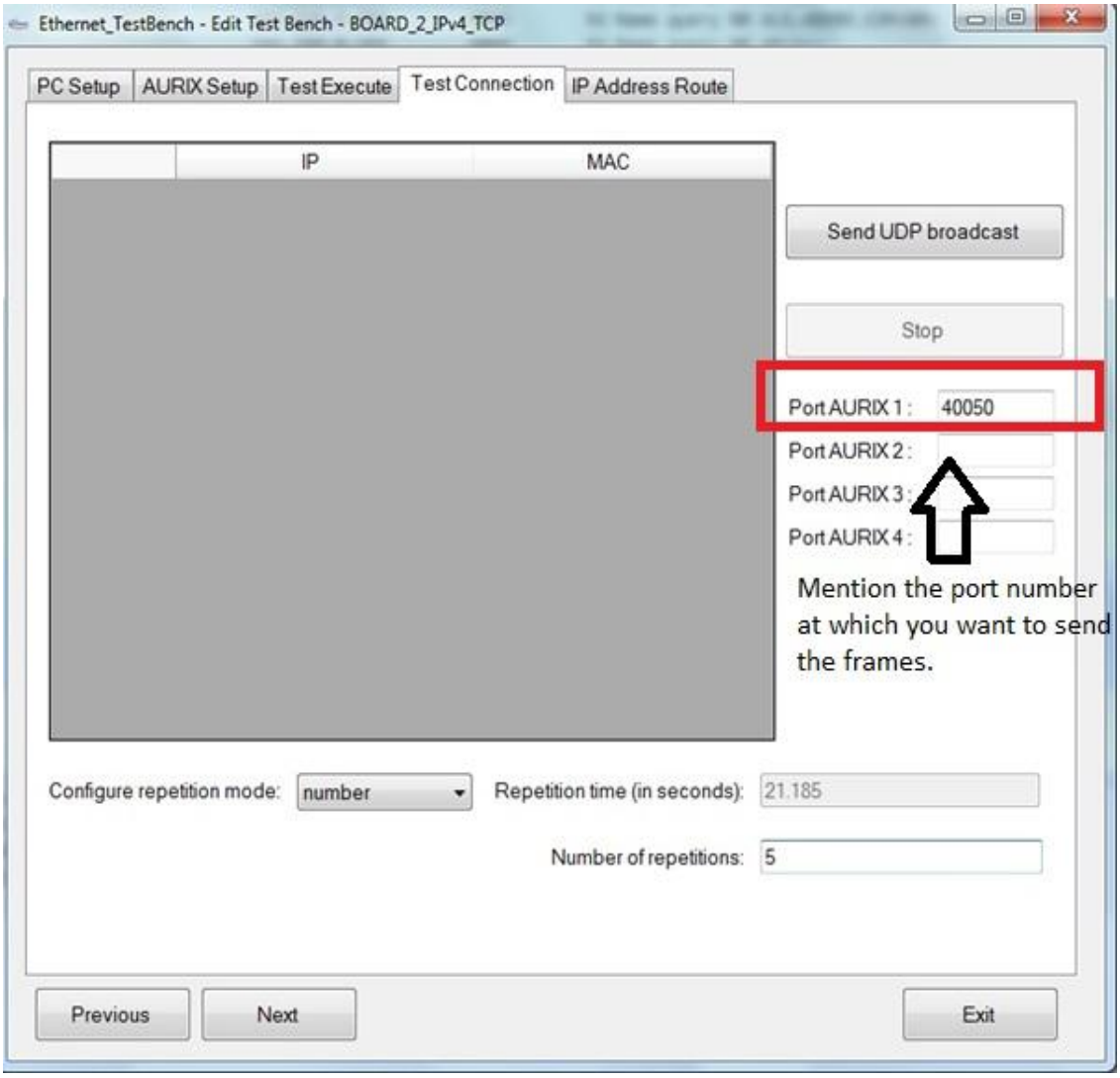
Screenshot 6.5: IP address route tab to send the ICMP packets to the board

(v) Then the TCP frames are tested using the Test Execute option of the test bench. Here at this moment click on the RUN option and it tends to make a TCP connection with the board. As success is achieved in creation of the TCP connection it additionally shows a success message within the text box as shown.



Screenshot 6.6: Test execute tab to send the TCP packets to the board

(vi) In the Test Connection section the UDP broadcast packets are tested to be sent to the board. Using the option Send UDP broadcast the test bench sends a series of UDP broadcast to the board.



Screenshot 6.7: Test connection tab to send the UDP broadcast packets to the board.

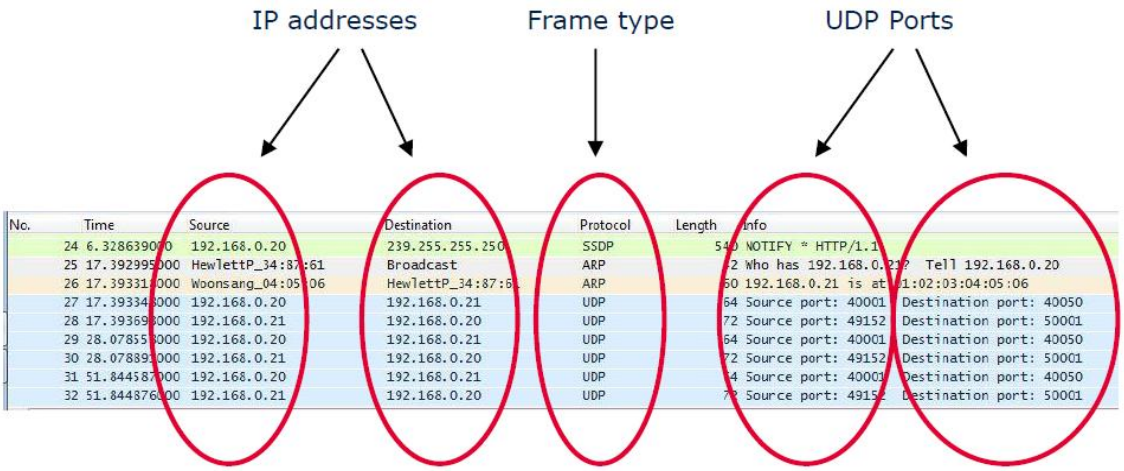
2. SIMULATION RESULTS

For testing the setup the port numbers that have been used for AURIX are as follows:

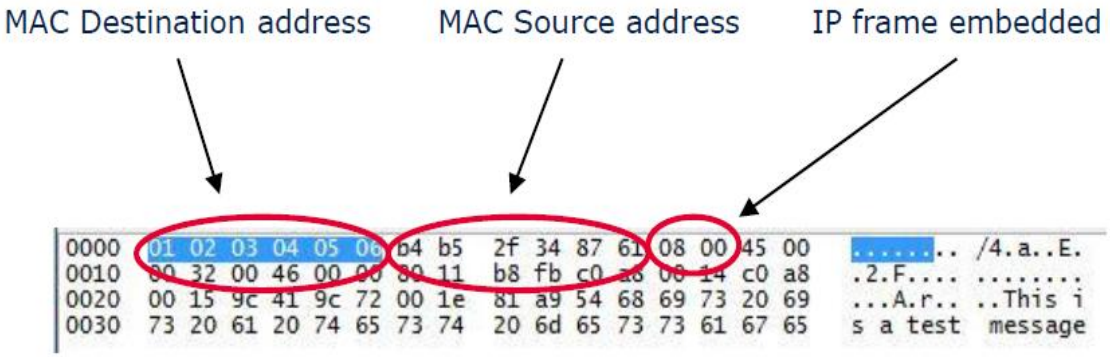
- PC port: 50001
- AURIX port: 40050

(a) Launch Wireshark, click on select your interface and click on “Start”

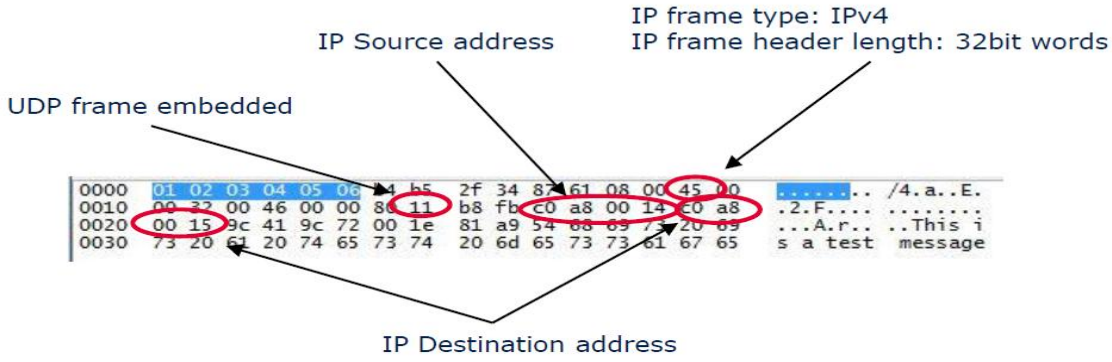
(b) Transferred frames checked with Wireshark



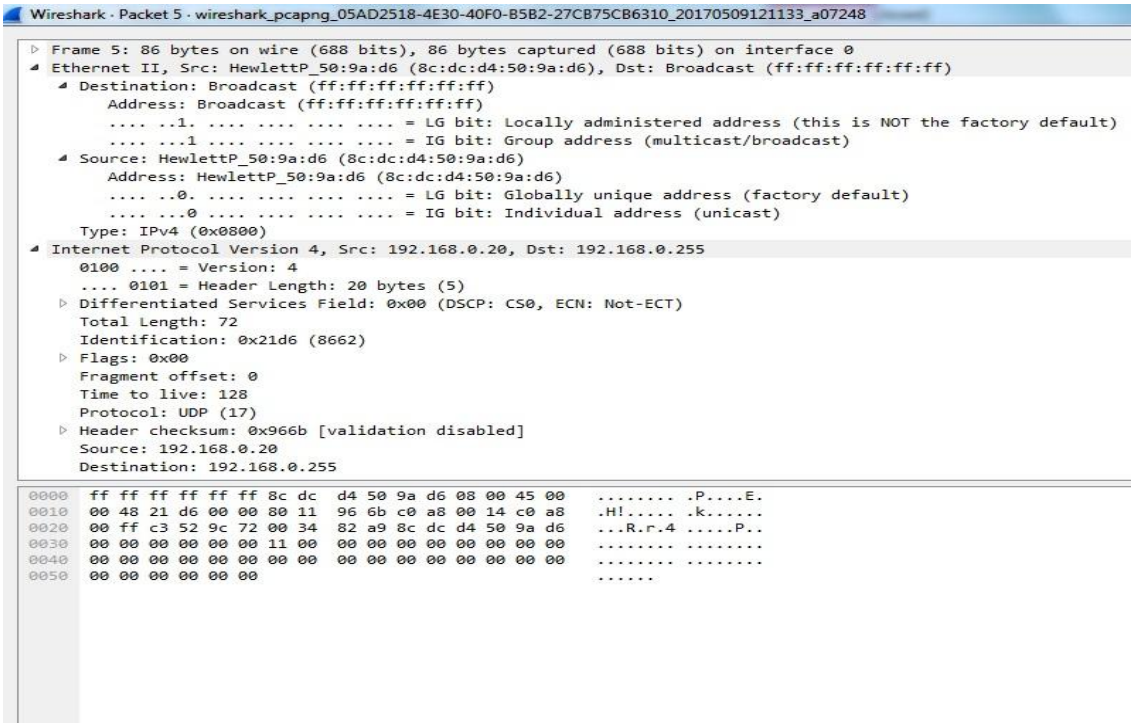
Screenshot 6.8: Various sections of the transferred frame



Screenshot 6.9: Ethernet header



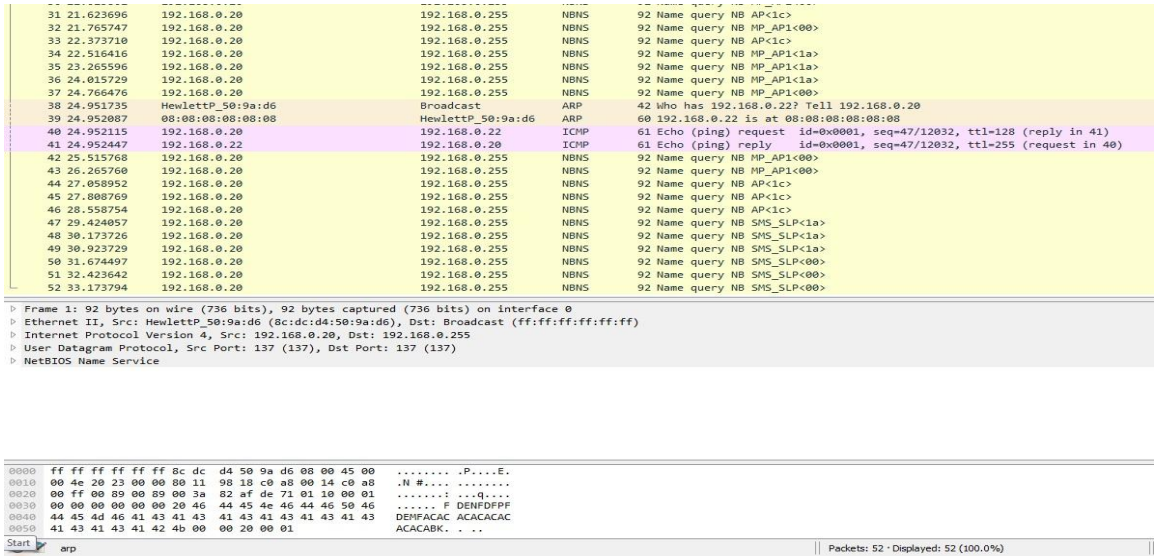
Screenshot 6.10: IP header



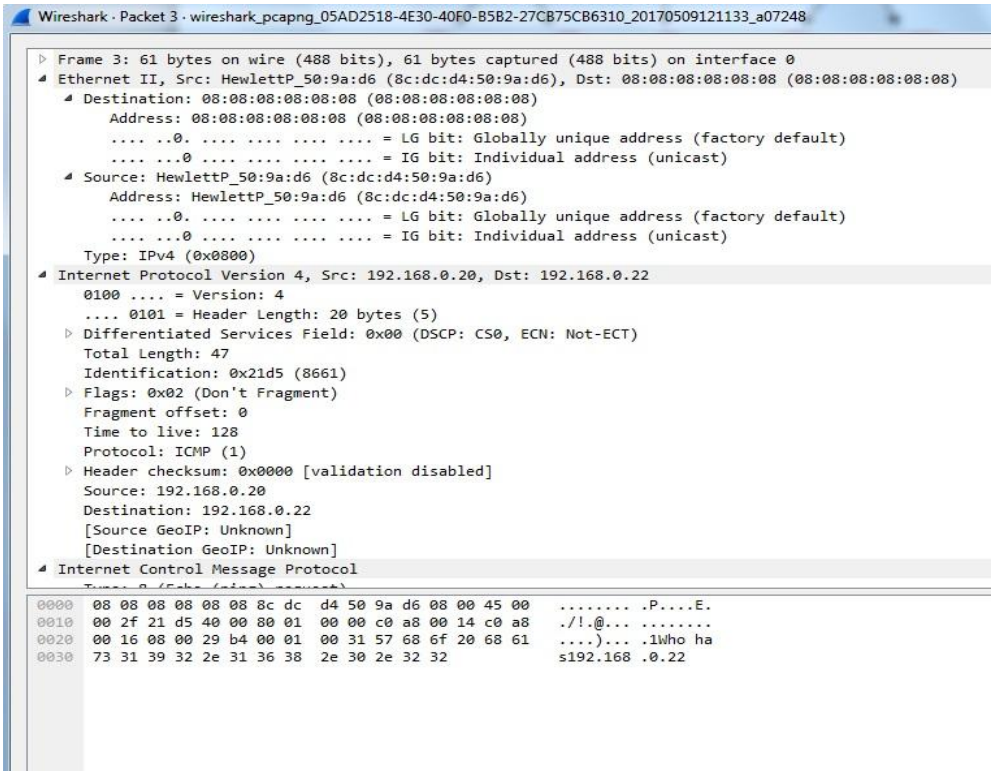
Screenshot 6.11: Transferred frame checked with Wireshark

(c) ICMP and ARP Results

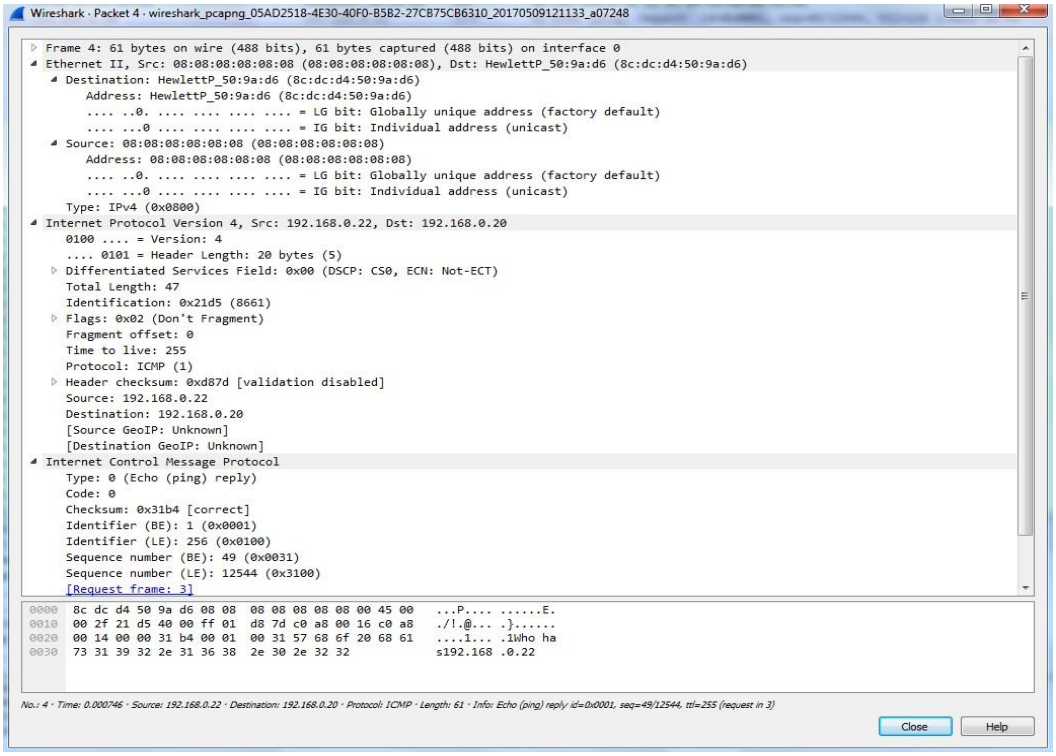
After the elf file is loaded onto the board then in order to validate the TCP, UDP and VLAN frames, ICMP and ARP packets are exchanged between the test bench application and the hardware in order to verify that an appropriate connection has been made.



Screenshot 6.12: Wireshark ICMP check



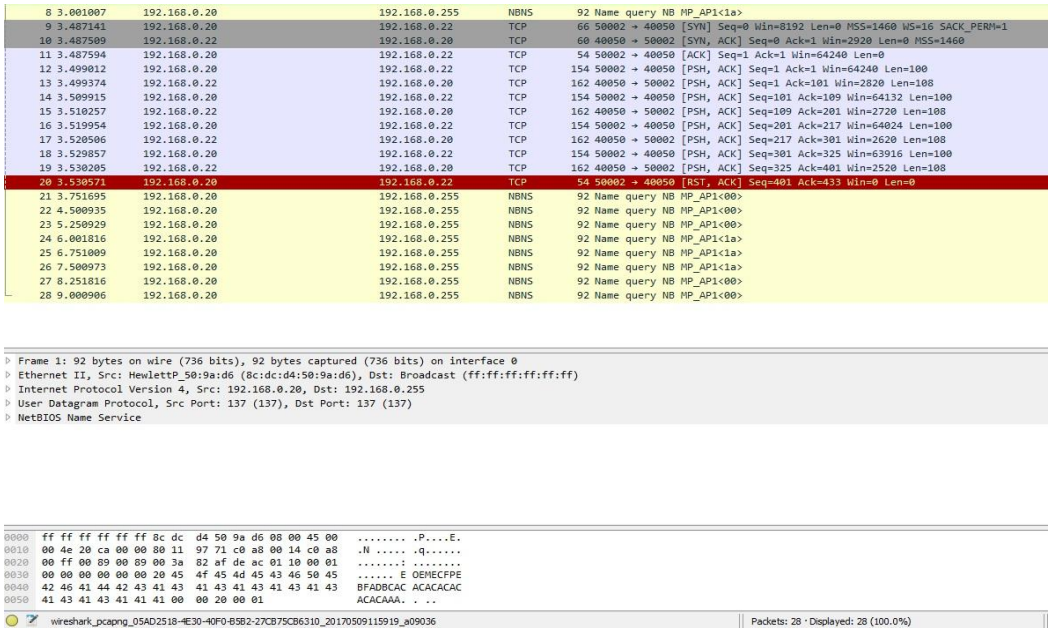
Screenshot 6.13: Wireshark view of ARP frame



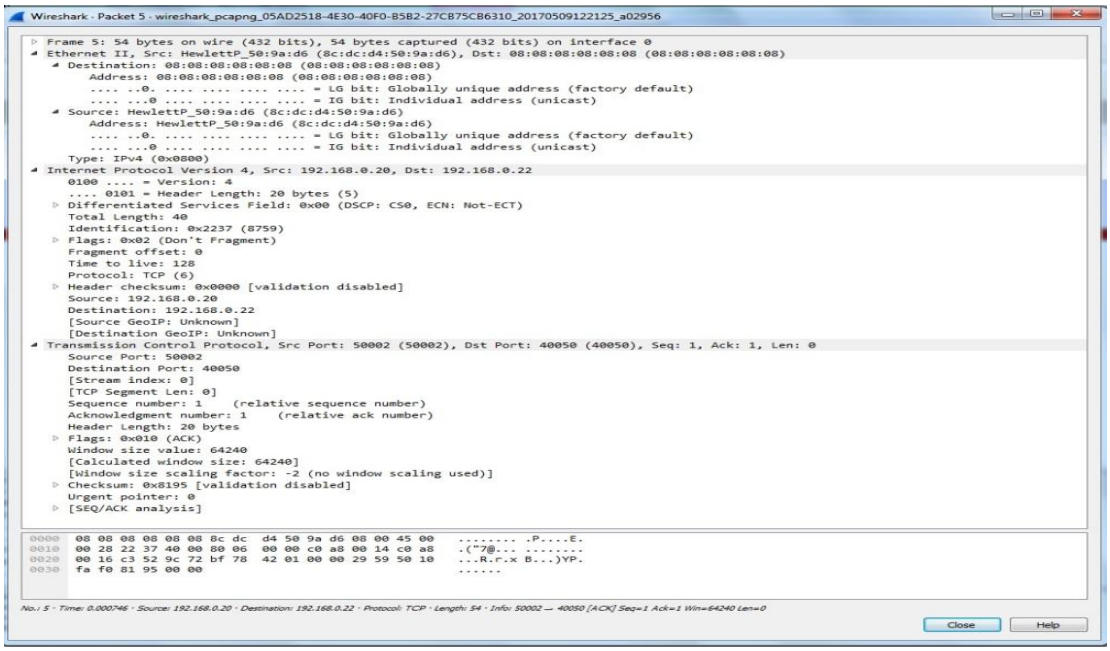
Screenshot 6.14: Wireshark view of ICMP frame

(d) TCP Results

Since TCP is a connection oriented protocol therefore, the TCP frames captured show the 5-way handshake between the test bench application and the hardware. It shows how the connection is established between the two and the sequence of data being exchanged.



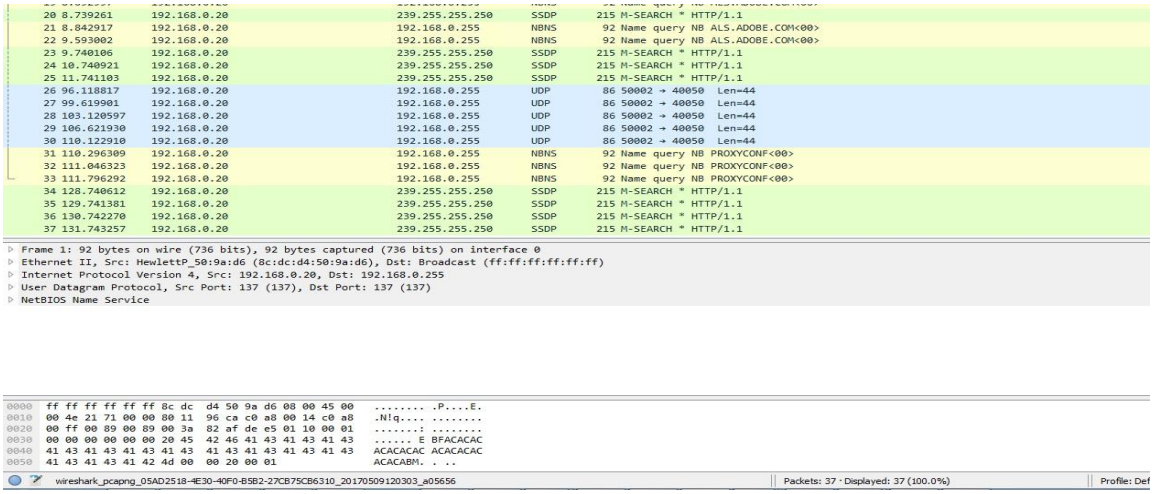
Screenshot 6.15: Wireshark TCP check



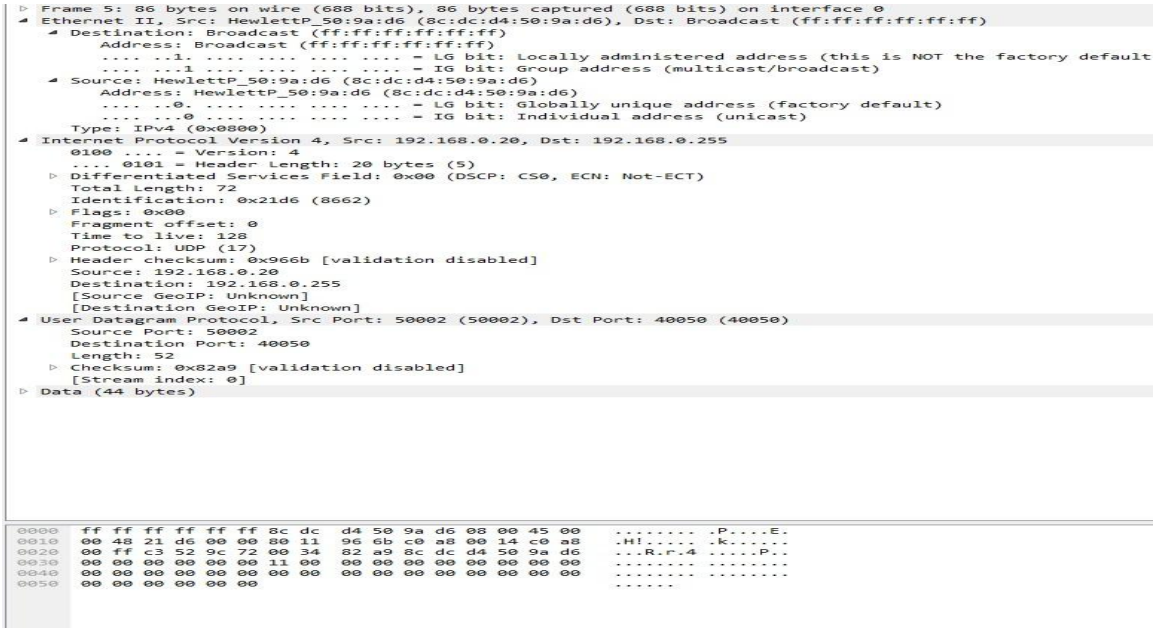
Screenshot 6.16: Wireshark view of TCP frame

(e) UDP Results

As UDP is a connection-less protocol therefore, the frames captured here show the UDP broadcast packets being sent from the test bench application to the respective network. This UDP validation is required for implementing SHE+ and such other protocols at the transport layer from a safety perspective.



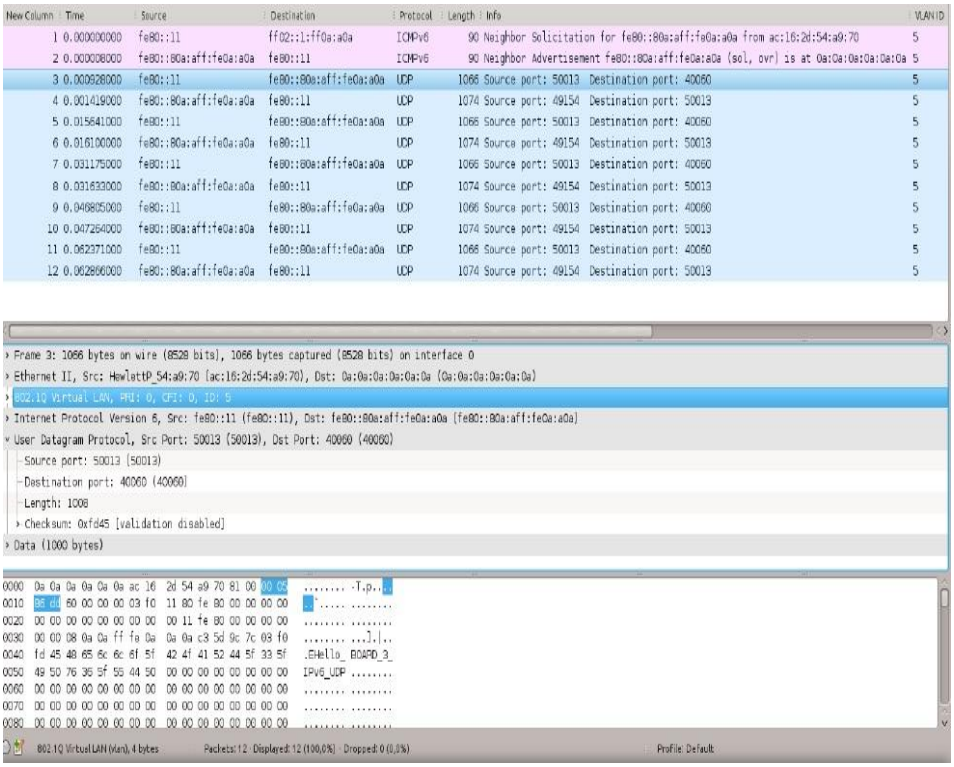
Screenshot 6.17: Wireshark UDP check



Screenshot 6.18: Wireshark view of UDP frame

(f) VLAN Results

VLAN frames show a UDP frame has a VLAN tag attached with it which ensures another level of security when the packet is being received at the other end. It shows the VLAN ID to which that packet has been sent along with the port numbers and the source and destination address.



Screenshot 6.19: Wireshark VLAN check

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

Automotive industry is going through tremendous change of trends, with the introduction of new technologies, like Advanced Driver Assistant System (ADAS), Smart Transportation, Fleet Management and Autonomous Driving systems. These are the cutting edge technologies which require connected cars with more complex intra network, Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I) and vehicle to Internet of Things (IoT) connectivity. Such complex connected ecosystem demands security on the network to protect safety critical ECUs from various network attacks as ECUs belonging to infotainment/telematics functional domains interact with external networks which becomes source of threats.

VLAN essentially permits us to have a view of multiple logical networks of a single physical ethernet (local area network). And therefore the foremost reason behind using this standard is to boost the network security by preventing the devices to access our LAN resources and secondly by doing this we can definitely improve the performance of our network.

7.2 Future Scope

In my research as I have implemented VLAN on AURIX platform (1G), there can be a scope of further enhancements which can be carried forward such as:

- It can further be implemented on 2G as well as 3G platform which are coming soon.
- Also we can also implement firewall along with VLAN in order to provide another level of security in order to communicate with the outside world as well as for in-vehicle communication. A firewall implemented in every VLAN can provide even more secure internal as well as external network to communicate.

REFERENCES

[1] Peter Hank, Steffan Muller, et al., "Automotive Ethernet: In-vehicle Networking and smart mobility," ACM, Sep.2013.

[2] R Simon, JS Jayasudha, et al., "Overview of Diagnostic over IP (DOIP), Ethernet Technology and Lighweight TCP/IP for embedded systems," International Journal of Advanced, 2013.

[3] Michele Selvatici, Massimiliano Ruggeri, Luca Dariz., "Advanced gateway services for real-time in-vehicle," IEEE, 2015.

[4] Kai Muller, Franz Korf, et al., "A Real-time Ethernet Prototype Platform for Automotive Applications," IEEE International Conference on Consumer Electronics, Berlin, 2011.

[5] Chung Wei Lin, Huafeng Yu, "Coexistence of safety and security in next generation Ethernet based Automotive networks," ACM, Austin, USA, 2016.

[6] Giorgio Malaguti, Massimo Dian et al., "Comparison on Technological Opportunities for In-Vehicle Ethernet Networks," IEEE, 2013.

[7] Catalin-Virgil Briciu, Ioan Filip "The Challenge of Safety and Security in Automotive Systems," 9th IEEE International Symposium on Applied Computational Intelligence and Informatics, Timişoara, Romania, 2014.

[8] Roland Rieke, Marc Seidemann et al., "Behavior Analysis for Safety and Security in Automotive Systems," 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, IEEE, 2017.

[9] Bernhard Jungk, "Automotive Security State of the Art and Future Challenges," IEEE, 2016.

[10] Tim Kiravau, Mikko Sarela, Jukka Mannar , "A Survey of Ethernet LAN Security," Communications Surveys & Tutorials, IEEE, 2013.

[11] Marko Wolf, Kerstin Lemke, Christof Par, "Secure In-Vehicle Communication" Embedded Security in Cars," in Springer , Berlin Heidelberg ,New York, 2006.

[12] Alexander Camek, Christian Buckl, Pedro Sebastio Corria , "An automotive Side-View system based on Ethernet and IP," 26th International Conference on

Advanced Information Networking and Applications Workshops, IEEE, 2012.

[13] Karl Koscher , Alexie Czeskis, Franziska Roesner , "Experimental Security Analysis of a Modern Automobile," Symposium on Security and Privacy, IEEE , May 2010.

[14] Hendrik Schweppe, Yves Roudier , "Security and privacy for in-vehicle networks," 1st International Workshop on In Vehicular Communications, Sensing, and Computing (VCSC), IEEE , 2012.

[15] Christopher Corbett, Elmar Schoch, et al., "Automotive Ethernet: Security opportunity or challenge," Lecture Notes in Informatics, 2006.

[16] Shane Tuohy, Martin Glavin et al., "Intra-Vehicle Networks: A Review," Transaction on intelligent transportation systems, IEEE, 2014.

[17] "TechTarget,"[Online].Available:http://searchsecurity.techtarget.com/tip/Popular-VLAN-attacks-and-how-to-avoid-them. [Accessed 2016].

[18] "SANS Institute," [Online]. Available: https://www.sans.org/reading-room/whitepapers/networkdevs/virtual-lan-security-weaknesses-countermeasures-1090. [Accessed 2016,2017].

[19] R.L.Bull,"ClarksonUniversity,"2015.[Online].Available: http://people.clarkson.edu/~bullrl/classes/CS708/bullrl_CS708_S15.pdf.[Accessed 2016].

[20] "VLAN,[Online].Available:http://www.engpaper.com/vlan-research-papers.htm. [Accessed 2016,2017].

[21] "RedScan," 2013 Oct. [Online]. Available: http://www.pages02.net/coastams-networkbox/newsletter/newsletter_oct2013_article1.html. [Accessed 2017].

[22] B.Gale,"IEEE,"2014.[Online].Available: https://standards.ieee.org/events/automotive/2014/19_Ethernet_Car_Security.pdf. [Accessed 2016,2017].

[23] "NetworkComputing,"[Online].Available: http://www.networkcomputing.com/networking/how-ethernet-can-secure-connected-car/1173922401/page/0/1. [Accessed 2016].

[24] M. M. e. al., "Automotive Ethernet: Security opportunity or challenge," Lecture Notes in Informatics, 2006.

[25] Oct2013.[Online].Available:http://www.pages02.net/coastams-networkbox/newsletter/newsletter_oct2013_article1.html. [Accessed 2017].

LIST OF PUBLICATIONS

1. Aashima Sharma, Sundar, Ponnarasu “Whitepaper on VLAN implementation with LwIP in automotive” *Infineon Technologies, ATV-Software*, 2017.
2. Rahul Singh, Aashima Sharma, Mansoor Ahmed “Whitepaper on Minimal TLS implementation with LwIP” *Infineon Technologies, ATV-Software*, 2017.
3. Aashima Sharma, Dr Maninder Singh, Sundar “VLAN in automotive” *International Institute of technology and research*, 2017.(Abstract accepted, paper under consideration)