A

PROJECT REPORT ON

COMMUNICATION BETWEEN RABBIT PROCESSOR AND PERSONAL COMPUTER USING TCP

SUBMITTED TO THE UNIVERSITY OF PUNE,
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING
(ELECTRONICS AND TELCOMMUNICATION)

BY

NIHAR S. BENDRE	B8433013
PRAMESH M. SABNE	B8433084
AMEY A. KULKARNI	B8433060

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION

STES'S
SINHGAD ACADEMY OF ENGINEERING

KONDHWA (BK),

PUNE 411048

SINHGAD TECHNICAL EDUCATION SOCIETY

CERTIFICATE

This is to certify that the project entitled

"COMMUNICATION BETWEEN RABBIT PROCESSOR AND PERSONAL COMPUTER USING TCP."

Submitted by

NIHAR S. BENDRE	B8433013
PRAMESH M. SABNE	B8433084
AMEY A. KULKARNI	B8433060

is a bonafide work carried out by them under the supervision of Prof.C.G.Patil and it is approved for the partial fulfilment of the requirement of University of Pune for the award of the Degree of Bachelor of Engineering (Electronics and Telecommunication).

This project report is not earlier submitted to any other Institute or University for the award of any degree or diploma.

Prof. C.G. Patil	Dr. K.P.Patil	Dr. A.G. Kharat
Guide	H.O.D.	Principal
Department of E&TC	Department of E&TC	SAE, Pune

Pune-48

Place: Pune

Date:

ACKNOWLEDGEMENT

It gives us great pleasure in presenting the project report of **"COMMUNICATION BETWEEN PC AND RABBIT PROCESSOR USING TCP".** With deep sense of gratitude we acknowledge the guidance of our project guide ***Prof. C. G. Patil.*** The time-to-time assistances and encouragement by him has played a vital role in preparation and functioning of our project which helped us in completing the project work in time. We will always remain indebted to him.

We are grateful to our Principal ***Dr. A.G. Kharat*** and Head of Dept. ***Dr. K.P. Patil***. We also take this opportunity to convey our sincere thanks to the teaching staff of Electronics and Telecommunication Dept.

Words are inadequate in offering our thanks to our Project Guide ***Prof. C. G. Patil,*** for his encouragement and cooperation in carrying out the project work. All our efforts and hard labour may have gone waste if we were not guided by him; he has guided us all the way and given us all his support. Besides completing such a project requires hard determination and positive energy and it is only possible if one is blessed by GOD.

It gives us a great pleasure to express my deep sense of gratitude and indebtedness to my guides **Mr. Suresh Sabhapathy** and **Mr. Abhay Bhumkar** for their valuable support and encouraging mentality throughout the project. We are highly obliged to them for providing me this opportunity to carry out their ideas and work during our project period and helping us to gain the successful completion of my Project.

We are highly grateful to the Honorable Centre Director of GMRT-NCRA, **Mr. Swarnakanthi Ghosh**; Chief Scientist **Mr. Yashwant Gupta**; and **Mrs. N.S.Deshmukh**, STP Coordinator, GMRT, for giving us this golden opportunity to be a part of this organization for this period.

We are also highly obliged to **Mr. J. K. Solanki**, Admin Officer D, for his support and valuable encouragement throughout my project.

Last but not the least our thanks also go to **all Admin Group**, for arranging the facility for us to make our stay at GMRT one of the most memorable period of our life.

We express our deep gratitude to all......

NIHAR S. BENDRE

PRAMESH M. SABNE

AMEY A. KULKARNI (B.E, E&TC)

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 - INTRODUCTION

1.1 Background-

The Tata Institute of Fundamental Research (TIFR) has set up two major radio astronomy facilities during the past 3-4 decades. One is at Ooty, about 300 kms south west of Bangalore; the second one is at Khodad, about 80 Kms north of Pune and about 100 Kms east of Mumbai (Bombay). While Ooty radio telescope has been in operation for the past 35 years, The Giant Metrewave Radio Telescope (GMRT) is a new facility commissioned in 2001, near Pune. We describe GMRT briefly and the solar observations being made with it.

Two major radio astronomy facilities of TIFR are at Ooty (ORT) and at Khodad (GMRT). While ORT has been in operation for 35 years, GMRT was opened as an international facility since 2001. We briefly describe only GMRT here.

The Giant Metre-wave Radio Telescope (GMRT) is a mammoth instrument for studying astrophysical phenomena at low radio frequencies (40 to 1600 MHz). This has been built in India by NCRA-TIFR as a national project at a cost of about 15 million US $ (1992). • The array telescope consists of 30 antennas of 45 metres diameter, each, operating at metre wavelengths -- the largest in the world at these frequencies!

Location and Configuration of the GMRT-

- ✓ Latitude : 19 deg N
- ✓ Longitude : 74 deg E
- ✓ About 70 km N of Pune
- ✓ 100 km E of Mumbai.
- ✓ 30 dishes; 45 m diameter
- ✓ 12 dishes in central compact array
- ✓ Remaining along 3 arms of a Y-array
- ✓ Total extent: 14 km radius, resolution of a 25 km size antenna is achieved!

Some important sub-systems of the GMRT-

- ✓ Mechanical sub-system
- ✓ <u>Servo sub-system</u>
- ✓ Antenna feeds (including positioning & control)
- ✓ Receiver chain -- analog
- ✓ Optical fibre sub-system
- ✓ Receiver chain -- digital
- ✓ Telemetry sub-system
- ✓ “On-line” Control and Monitor sub-sytem
- ✓ Off-line data processing chain(s)

SERVO System of GMRT controls and monitors the movement of the 30 antennas in Azimuth and Elevation. We all know that there 30 fully steerable parabolic dishes of 45 metre diameter each at GMRT. Motion of these gaint antennas need to be controlled by a precession control system. Pointing of the antennas should be accurate i.e. the radio source, antenna focused and the antenna center should be aligned. The GMRT servo system has designed with three nested control loops to achive the pointing accuracy of (1 or 2) arc minutes RMS for wind speed less than 20 km/ph. Because of high weight alt-azimuth mount is most favorable approach for positioning the dish antenna. Here the elevation axis sits on the azimuth drive. The elevation drive moves antenna up and down directions while azimuth drive moves antenna in clockwise & counter-clockwise direction. Hence, enabling the antenna to point anywhere in the sky. There are also various specifications of Servo System at GMRT such as Dish Mount – Dish Movement – Dish speed specifications, Tracking maximum and minimum speed specification, Accuracy Specifications, Operating Voltage specification, Design and Survival wind speed specification, etc.

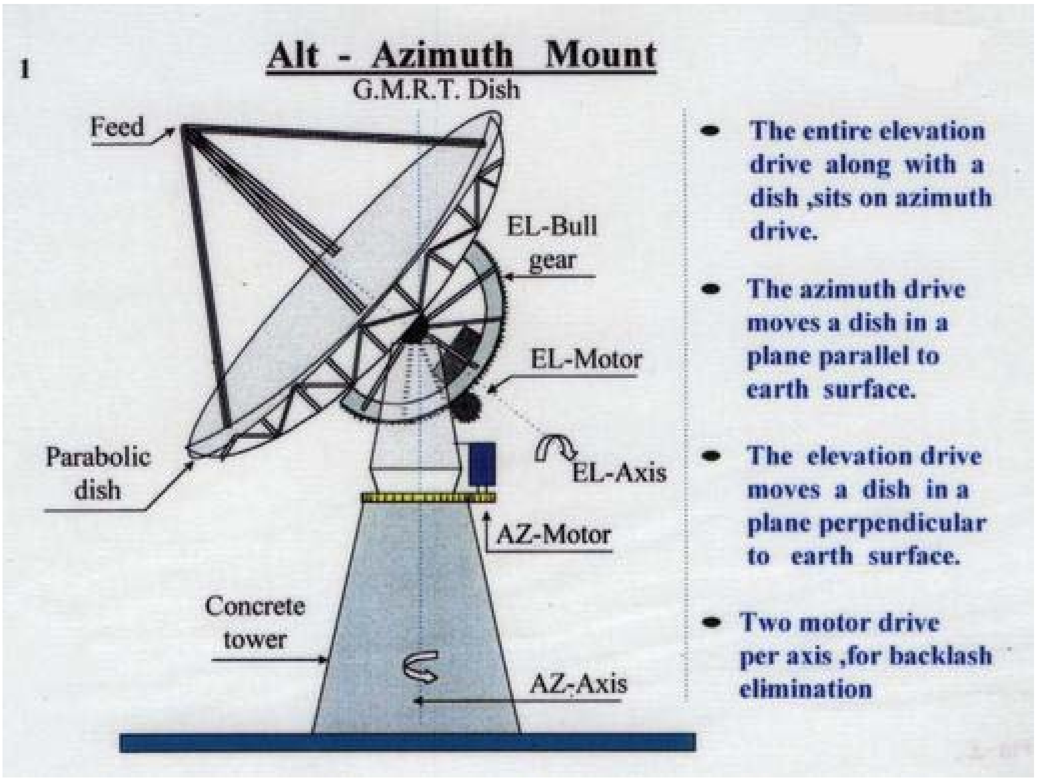

1
Alt - Azimuth Mount
G.M.R.T. Dish
Feed
EL-Bull gear
EL-Motor
EL-Axis
Parabolic dish
AZ-Motor
Concrete tower
AZ-Axis
The entire elevation drive along with a dish ,sits on azimuth drive.
The azimuth drive moves a dish in a plane parallel to earth surface.
The elevation drive moves a dish in a plane perpendicular to earth surface.
Two motor drive per axis ,for backlash elimination

Figure 1 Function of GMRT Servo System

Some key features of the Control and Monitor Systems are-

1) The rotation of all the thirty antennas in AZIMUTH and ELEVATION through SERVO CONTROL COMPUTER (SCC).

Azimuth--> -270 TO +270 deg Elevation --> 17 TO 90 deg

--> Using ABC - SERVO RS422 communication link @ 9.6 Kbps.

2) Alignment of the required feed for observation to the focus of the dish thro' Feed Position System (FPS).

0 deg --> 233/610 MHz 90 deg --> 150 MHz 180 deg --> 1420 MHz 270 deg --> 325 MHz

--> Using ABC - FPS RS485 communication link @ 9.6 Kbps.

3) Selection of FRONT END SYSTEM parameters like observing freq. band, Noise Cal, Channel Swap etc through **MCM '5'** & **MCM - FE** interface card in the Common Box.

4) Sets the LO Freq. , IF bandwidth (6,16,32 MHz),IF attn., ALC ON/OFF etc. using **MCM-s '2', '3' & '9'** located in LO/IF SYSTEM

5) Sets Baseband bandwidth (62.5 kHz to 16 MHz) using MCM-s located in BASEBAND SYSTEM through **Antcom** '0'.

6) Monitors the C & M system parameters at Antenna shell using **MCM '0'** at NRR and at CEB using 6 MCM-s through **Antcom '31'**.

7) Monitors the Temperature at various points in RFI cage in each antenna through Temperature Monitor system using **MCM '0'**.

In addition, it also provides the vital voice communication link between CEB & all the Antennas i.e. You can contact anybody in any antenna using C & M SYSTEM.

The communication medium is a single mode analog optical fibre link operating @ 1310 nm between CEB and all the antennas i.e. two fibres are available for each antenna. So,the digital command & control signals are converted to analog form using non-coherent FSK techniques and sent along with the LO and IF signals. In the forward link,Telemetry command signals are combined with the LO carriers and transmitted to all the antennas and in the return link,Telemetry monitor signals are combined with the IF signals and brought to CEB. The FORWARD link uses 18 MHz carrier & the RETURN link uses 205.5 MHz.

Operating Frequencies of GMRT-

The operating frequencies of GMRT are as follows,

SERIAL NUMBER	FREQUENCIES
1	**40-60MHz**
2	**120-180MHz**
3	**225-245MHz**
4	**300-360MHz**
5	**580-650MHz**
6	**1000-1430MHz**

Table 1 Operating Frequencies of GMRT

Science with GMRT-

- ✓ Sun and Solar System astronomy
- ✓ Study and discovery of Pulsars
- ✓ Studies of Galactic Centre regions
- ✓ Discovery of organic molecules like Acetaldehyde
- ✓ Nearby galaxies, Dwarf Galaxies, etc
- ✓ Peculiar radio galaxies
- ✓ Red shifted hydrogen line studies
- ✓ Gamma Ray bursters, etc.

1.2 History-

Rabbit Semiconductor is the company which designs and sells the Rabbit family of microcontrollers and microcontroller modules. For development, it provides Dynamic C, a non-standard dialect of C with proprietary structures for multitasking. Rabbit Semiconductor was formed expressly to design a better microprocessor for use in small- and medium-scale single-board computers. The first microprocessors were the *Rabbit 2000* and the *Rabbit 3000*. The latest microprocessor is the *Rabbit 4000*. Rabbit microprocessor designers have had years of experience using Z80, Z180, and HD64180 microprocessors in small single-board computers. The

Rabbit microprocessors share a similar architecture and a high degree of compatibility with these microprocessors, but represent a vast improvement.

Rabbit Semiconductor was purchased in 2006 by Digi International. Before the purchase, Rabbit Semiconductor was a division of Z-World, Inc. Z-World developed and manufactured embedded controller products as well as embedded software development environments.

Giant Metrewave Radio Telescope (GMRT), located near Pune in Maharashtra, India, is the world's largest array of radio telescopes at metre wavelengths. It is operated by the National Centre for Radio Astrophysics, administered by Tata Institute of Fundamental Research, Mumbai. A huge antenna dish (like the antenna that the cable-operator has on his roof) is moving atop a stationary structure. It is not a mean task for a 45 metre antenna to move mechanically. It is awesome. It is beautiful. Stretching my eyes over the horizon, I saw few more antenna - all tilting together searching for a common goal. Located on the quite and isolated of Pune, is Khodad (near Narayangaon) where I'm standing. What I'm watching is the Giant Metrewave Radio Telescope (GMRT) of the Tata Institute of Fundamental Research, Mumbai the world's largest radio telescope. It consists of thirty45-m dishes arranged in a Y-shaped configuration spread over distances of ~25 km. Work on GMRT was started ~ 1989 under the leadership of Prof. Govind Swarup and by 1996,all the 30 antennas were operational. The telescope is being used by astronomers from all over the world to study and learn more about astronomical objects emitting in radiofrequencies. GMRT works in the radio regime where the wavelength is of the light is order of a metre (the wavelength of the red light that you see is about a million times smaller than a metre).There are number of astronomical objects which emit mainly in the radio wavelengths but only the powerful emitters can be detected since these objects are very distant. One of the science projects of GMRT is to detect hydrogen from very distant galaxies. Hydrogen forms a major constituent of our Universe from which galaxies are formed. Atomic hydrogen emits radio emission at 21 cm. It is believed that the Universe is expanding and the distant galaxies are moving away further from us. GMRT was a huge project, and a lot of research went into it. Interestingly, the wired net used to make the antenna-dishes is an innovative technology by the Indian engineers. GMRT is one of the most challenging experimental programmes in basic sciences undertaken by the Indian scientists and engineers. The antennas have stopped moving and started collecting radio waves from the source.

Borland C++ is a C and C++ programming environment (that is, an integrated development environment) for MS-DOS and Microsoft Windows. It was the successor to Turbo C++, and included a better debugger, the Turbo Debugger, which was written in protected mode DOS.

Object Windows Library (OWL): A set of C++ classes to make it easier to develop professional graphical Windows applications.

Turbo Vision: A set of C++ classes to create professional applications in DOS. Those classes' mimics some of the aspects of a Windows application like: dialog boxes, messages pumps, menus, accelerators, etc.

Version History-

- ✓ Borland C++ 2.0 - (1991, MS-DOS)
- ✓ Borland C++ 3.0 - (1991) new compiler support to build Microsoft Windows applications.
- ✓ Borland C++ 3.1 - (1992) Introduction of Windows-based IDE and application frameworks (OWL 1.0, Turbo vision 1.0)
- ✓ Borland C++ 4.0 - (1993, Windows 3.x) MS-DOS IDE supported no longer, included OWL 2.0.
- ✓ Borland C++ 1.0 - (1992, OS/2)
- ✓ Borland C++ 1.5 - (?, OS/2)
- ✓ Borland C++ 2.0 - (1993, OS/2) Support for 2.1 and Warp 3. OWL 2.0. Included IBM SMART Toolset for automatically migrating Windows applications to OS2. Last version.
- ✓ Borland C++ 4.01
- ✓ Borland C++ 4.02 - (1994)
- ✓ Borland C++ 4.5
- ✓ Borland C++ 4.51
- ✓ Borland C++ 4.52 - (1995) Official support for Windows 95, OWL 2.5
- ✓ Borland C++ 4.53
- ✓ Borland C++ 5.0 - (1996, Windows 95) Released in March 1996. Works on Windows 95 and Windows NT 3.51. It does not (officially) work on Windows

NT 4.0 (which was still in development at that time). 3rd party tests exhibited some problems on NT 4.0. It does not work in Windows 3.x or DOS. Despite that, it can produce Win32, Win16 or DOS programs.

- ✓ Borland C++ 5.01
- ✓ Borland C++ 5.02 - (1997) Final independent release of the Borland C++ IDE (subsequently replaced up by the C++Builder series), final release to support compilation to (real-mode) MS-DOS target. Windows NT 4.0 officially supported.
- ✓ Borland C++ Builder 4.0 + Borland C++ 5.02 - (1999) Bundle combination to facilitate the migration to C++Builder.
- ✓ Borland C++ 5.5 - Command-line compiler only (not with IDE).

1.3 Present Scenario-

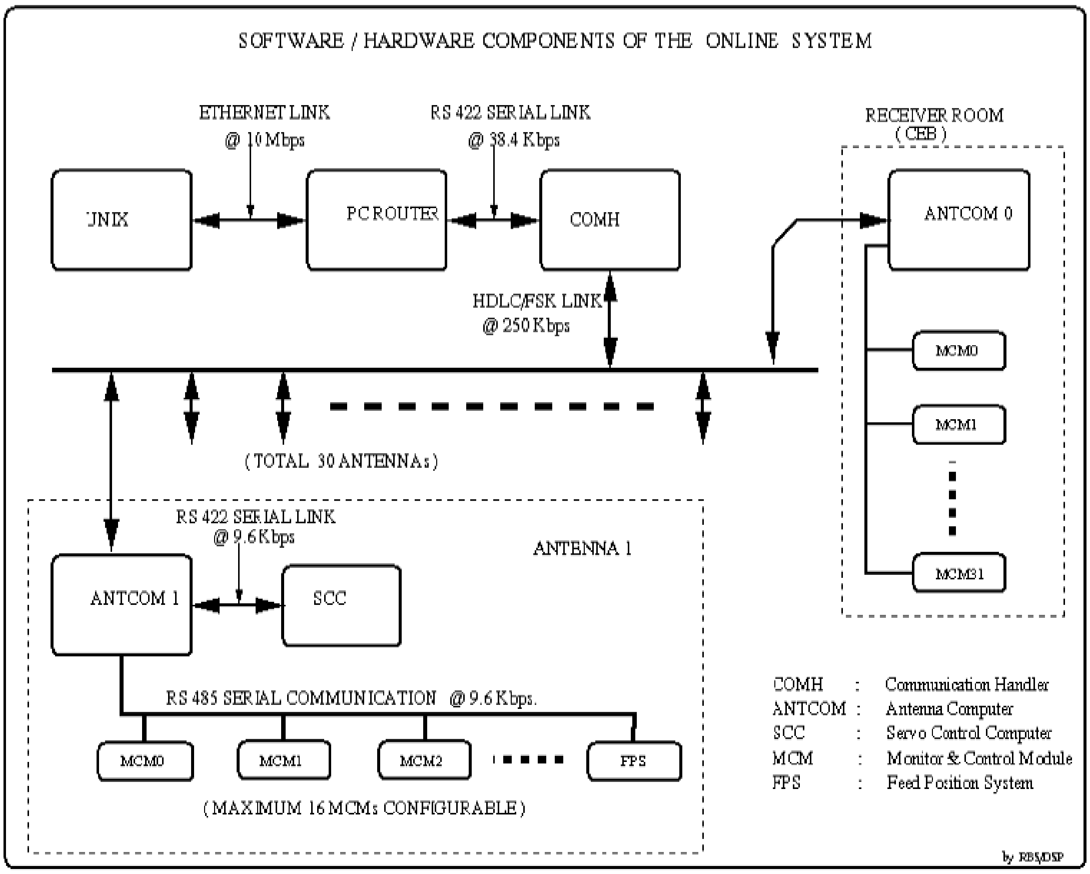

SOFTWARE / HARDWARE COMPONENTS OF THE ONLINE SYSTEM
ETHERNET LINK @ 10 Mbps
RS 422 SERIAL LINK @ 38.4 Kbps
RECEIVER ROOM (CEB)
UNIX
PC ROUTER
COMH
ANTCOM 0
HDLC/FSK LINK @ 250 Kbps
MCM0
MCM1
MCM31
(TOTAL 30 ANTENNAs)
RS 422 SERIAL LINK @ 9.6 Kbps
ANTENNA 1
ANTCOM 1
SCC
RS 485 SERIAL COMMUNICATION @ 9.6 Kbps.
MCM0
MCM1
MCM2
FPS
(MAXIMUM 16 MCMs CONFIGURABLE)
COMH : Communication Handler
ANTCOM : Antenna Computer
SCC : Servo Control Computer
MCM : Monitor & Control Module
FPS : Feed Position System
by RBS/DSP

Figure 2 Block Diagram Present System

The system is under development, present system of GMRT is based on Microcontroller 8051 and Microprocessor 8086.Here the serial communication between µc8051 and µp8086 is implemented by following the protocol RS232.So by using RS232 protocol of serial communication provides low data transfer speed.

At the base of each antenna there is one control room and one microprocessor 8086 card. The function of the up 8086 is to accept the instruction from uc 8051 to rotate the corresponding Servo Motor connected to antenna. Microcontroller 8051s are connected to each of up8086 and all the uc8051s are connected to one central control room at the main building. If any Astronaut wants to rotate antenna for certain angle for his observations, then he conveys his message to people in the telemetry sections. The people from the telemetry section send the control signal to uc8051.Microcontroller 8051 will receive the control message and take control action to send message to up8086.Then up8086 will receive that message and send control signal to servo motor to rotate the motor in the desired angle. Whole communication is done by following the serial communication protocol RS232.

1.4 Relevance-

The effective and efficient communication encounters the problems such as supported data rate, interfacing with the various electrical machines, compatibility with the new devices and existing protocols, difficulties in soft programming, etc. *Giant Metrewave Radio Telescope, National Centre for Radio Astrophysics, Tata Institute of Fundamental Research, Pune* has decided to upgrade their present Communication System to overcome these limitations.

TIFR has set up a unique facility for radio astronomical research using the metrewavelengths range of the radio spectrum, known as the Giant Metrewave Radio Telescope (GMRT). GMRT consists of 30 fully steerable gigantic parabolic dishes of 45m diameter each spread over distances of up to 25 km. GMRT aims to search for and study rapidly-rotating Pulsars in our galaxy and is one of the most challenging experiments for Indian scientists.

GMRT Khodad, Narayangaon is a well known research institute. It is a joint venture of Tata Institute of Fundamental Research and the Govt. of India. The institute specializes in the research in the field of Astronomy and Astrophysics. It is frequently visited by Scientists all over the world. This has the facility to tune antennas to observe galaxies, pulsars, quasars etc at certain altitude. This is done by focusing the antenna to the required altitude, continuous scanning of frequencies and processing of data. Hence the system should support very high data rate. But protocols implemented in the present system are based on old technologies; therefore, the data rate is low. It has been decided to use TCP instead of serial protocols such as RS232, RS422, and RS485 because of supported data rate, and compatibility with the new devices. Also when programming the processors likes µc8051 and µp80186, assembly language programming gets difficult and very lengthy.

The present system is working continuously in the Dynamic environment i.e. depending on the requirement of Scientists. The electrical machines like motors, etc have to be interfaced with the processors to control the rotation of motors and ultimately antennas. It is often seen that while interfacing the newly introduced electrical machine to the present system, system fails to support. Therefore, it has been decided to upgrade system by using the Rabbit processors which overcome the above discussed problems like low supported data rate, interfacing with the various electrical machines, compatibility with the new devices and existing protocols, difficulties in soft programming, as programming language for the processor is ‘C’.

The Rabbit Semiconductor was established majorly to design a better microprocessor for use in small- and medium-scale single-board computers. The Rabbit microprocessors share a similar architecture and a high degree of compatibility with current Microprocessors but show much better performance. The Rabbit 4000 is a high performance microprocessor with low electromagnetic interference (EMI), and is designed specifically for embedded control, communications, and network connectivity. Extensive integrated features and glueless architecture facilitate rapid hardware design, while a C-friendly instruction set promotes efficient development of even the most complex applications.

1.5 Block Diagram-

1.5.1 Proposed System

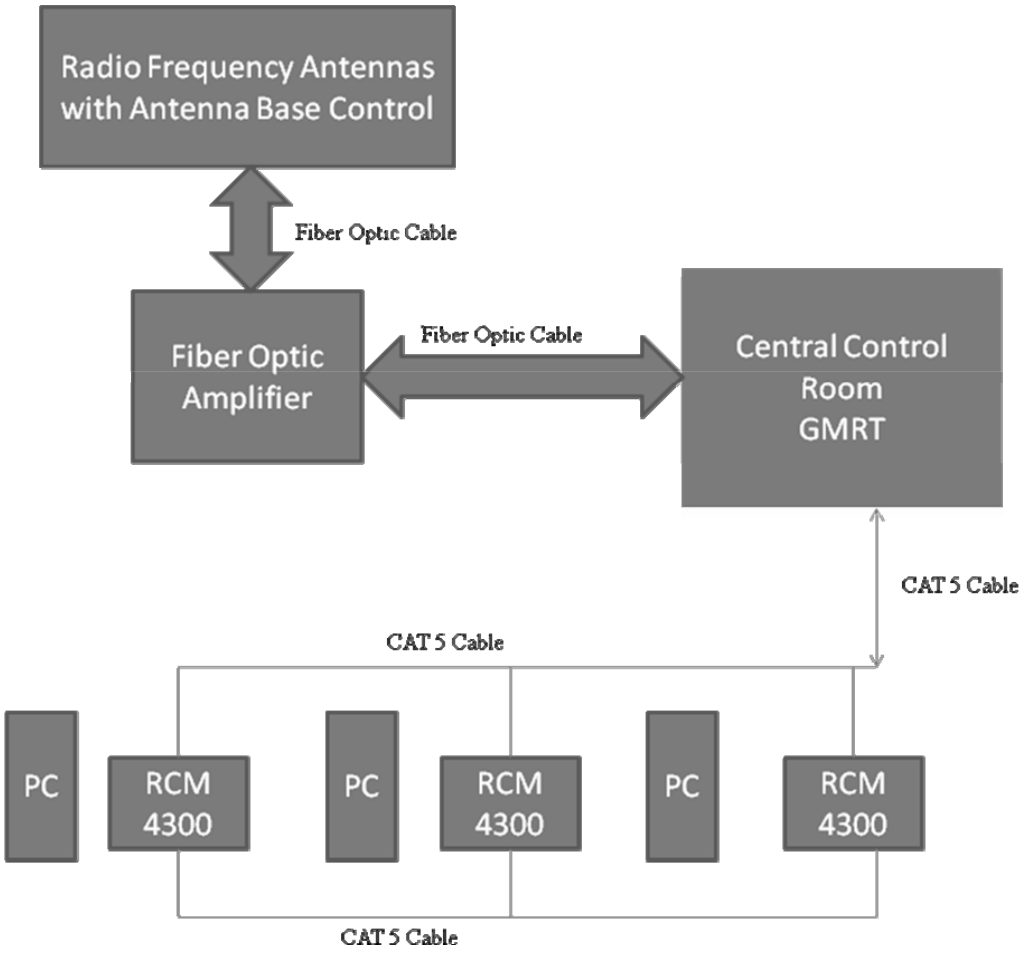


Figure 3 Block Diagram of Proposed System I

Here the devices like Personal Computer, RCM4300 are the main devices in the system. Role of personal computer are to send the instruction to RCM4300 processor. The programmer will decide the instruction for RCM4300 for each task.CAT5 cable (Ethernet Cable) is used as a transmission line which connects PC to RCM4300.Here the Personal Computer which we are using has Pentium 4(P4) processor inside. As we have RCM4300 processor which supports 10BaseT Ethernet, means it supports Ethernet of 10Mbps, so we are using CAT5 cable.CAT5 cable which also called as LAN cable is also capable of supporting this much speed.RCM4300 is the heart of the system, which accepts the instructions from PC, interprets them and takes corresponding control action on the instructions. So it will be configured as Transreceiver, as it receives the signal from PC and send control signal to other devices. For better understanding of the block diagram, we have created the same diagram in 3-D as well.

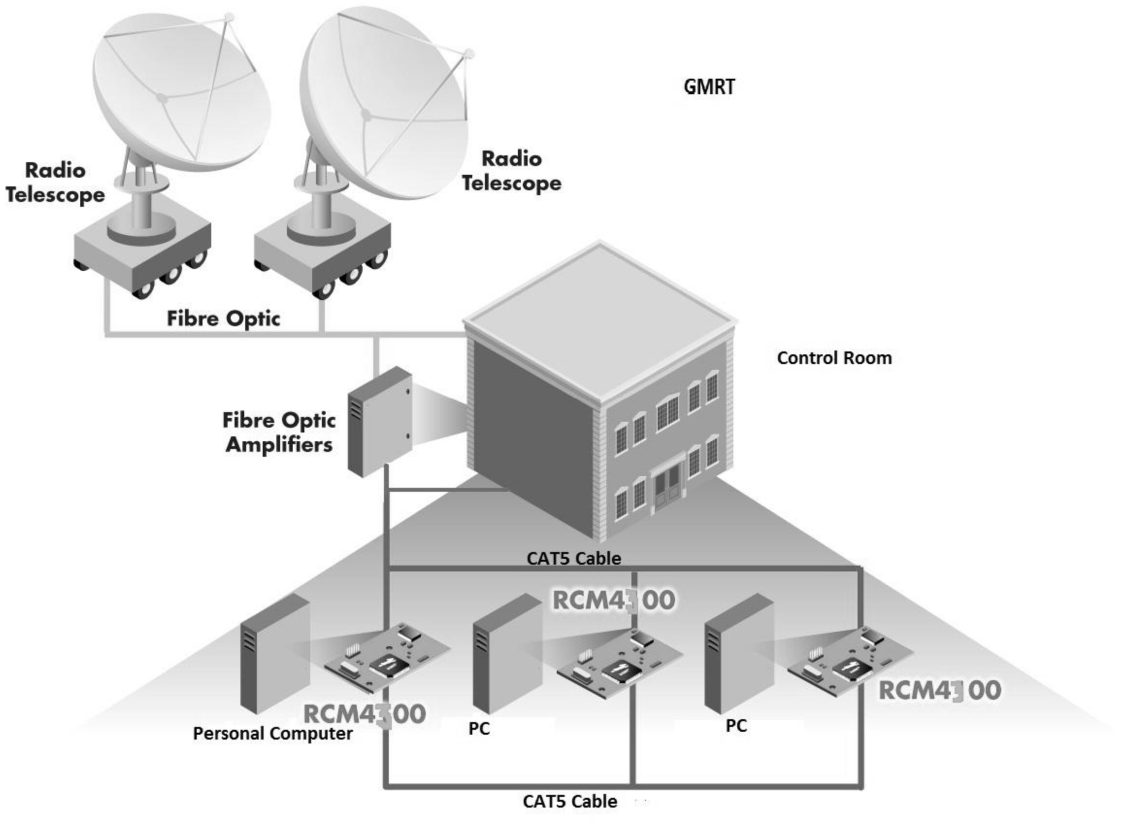


Figure 4 Block Diagram of Proposed System II (3- D)

1.6 Organization of the Report –

The remaining report is organized as, chapter [2] proceeds with the Literature Survey. Chapter [3] explains Ethernet Significance; it includes TCP layers, function of each layer in TCP, connection, frame formats and comparison of TCP with other protocols like RS 232 and RS 485, etc. Chapter [4] discusses System Description; it includes advantages of proposed system using Rabbit as well as TCP, also discuses the drawbacks and some of the applications.

The Hardware Description is explained in Chapter [5] which includes brief description about hardware components required to design this system like Rabbit 4000 Module, CAT 5 cable, Personal computer. Also performance comparison and specifications of CAT 5 cable are mentioned. Software Description is elaborated in Chapter [6]; it includes all the required software for this system like Dynamic C, Borland C, and Command Prompt. System Performance Evaluation is discussed in Chapter [7] which includes Flows of Transmitter and Receiver along with the problems faced while designing this system with their solution.

Chapter [8] explains Future scope of the system and chapter [9] tells the total costing of the system. Chapter [10] concludes the paper and at last References are mentioned from where we took help to design the system.

CHAPTER 2 - SURVEY

2.1 Literature Survey-

The architecture of this system is useful in a number of different areas. The literature survey for this system started from collecting the information about the existing systems for based on Rabbit Processors. We then divided the whole architecture in different blocks and studied each separately. We divided the system architecture like two of our group members will study the ports, registers present on the Rabbit Processor, and 1 group member will study related to TCP/IP, how to initiate Ethernet in the Rabbit.

The basic study included: The study of present GMRT system, the study of Rabbit Processor, TCP/IP [Ethernet protocol], how to initiate this protocol in the Rabbit Processor, study of Dynamic C, Borland C, and study of CAT5 cable and how to make it cross or straight, etc.

The present system of GMRT is based on Microcontroller 8051 and Microprocessor 8086.Here the serial communication between µc8051 and µp8086 is implemented by following the protocol RS232.So by using RS232 protocol of serial communication provides low data transfer speed.

The problems faced by GMRT are supported data rate, interfacing with the various electrical machines, compatibility with the new devices and existing protocols, difficulties in soft programming, etc. Hence, there is a need of upgradation.

Selection criteria for Rabbit Processor are, the Rabbit Semiconductor was established majorly to design a better microprocessor for use in small- and medium-scale single-board computers. The Rabbit microprocessors share a similar architecture and a high degree of compatibility with current Microprocessors but show much better performance. The Rabbit 4000 is a high performance microprocessor with low electromagnetic interference (EMI), and is designed specifically for embedded control, communications, and network connectivity. Extensive integrated features and glueless architecture facilitate rapid hardware design, while a C-friendly instruction set promotes efficient development of even the most complex applications.

For the study of TCP/IP, we used the reference book viz. *Behrouz A Forouzan ,"Data Communications and Networking", Third Edition, McGraw Hill Publication.* This book gave us the brief introduction of Data communication and Networking takes place through TCP. It also helped in studying TCP layers in details.

The **Transmission Control Protocol** (**TCP**) is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components of the suite, complementing the Internet Protocol (IP), and therefore the entire suite is commonly referred to as *TCP/IP*. TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. TCP is the protocol that major Internet applications such as the World Wide Web, email, remote administration and file transfer rely on. Other applications, which do not require reliable data stream service, may use the User Datagram Protocol (UDP), which provides a datagram service that emphasizes reduced latency over reliability. The TCP/IP model is a descriptive framework for computer network protocols created in the 1970s by DARPA, an agency of the United States Department of Defense. The name derives from the two most important protocols of the networking protocol suite, the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The model evolved from the operational principles of the ARPANET, which were an early wide area network and a predecessor of the Internet. The TCP/IP model is formalized in the Internet protocol suite and is sometimes called the *Internet model* or the *Do D model.*

The TCP/IP model describes a set of general design guidelines and implementations of specific networking protocols to enable computers to communicate over a network. TCP/IP provides end-to-end connectivity specifying how data should be formatted, addressed, transmitted, routed and received at the destination. Protocols exist for a variety of different types of communication services between computers.

TCP/IP has four abstraction layers as defined in RFC 1122. This layer architecture is often compared with the seven-layer OSI model; using terms such as *Internet reference model*, incorrectly, however, because it is descriptive while the OSI model was intended to be prescriptive, hence being a reference model.

The TCP/IP model and related protocols are maintained by the Internet Engineering Task Force (IETF).

For the study of **Rabbit 4000**, we used *Rabbit 4000 User Manual*. The Rabbit processor family shares many features with the Zilog Z80/Z180 processors. For example, the registers of a Rabbit 2000/3000 processor are almost the same as the registers of a Z80/Z180 processor. The Rabbit 4000 processor expands to include the use of 32-bit registers. The instruction set of Rabbit processors also closely resembles the instruction set of the Z80/Z180 family. While the opcodes of many instructions are the same between the Rabbit 2000/3000 processors and Z80/Z180 processors, the two families of processors are not binary compatible.

The Rabbit processor family has unique features. For example, the Z80/Z180 family disables interrupts once an interrupt is serviced by an interrupt service routine. However, the Rabbit processors permit interrupts to interrupt service routines according to priorities (a total of 4). As with the Z80/Z180 family, the Rabbit processors are CISC processors, as opposed to RISC competitors like the Atmel AVR processors. A comparison of clocks per instruction of the Rabbit processor against a typical RISC processor like the AVR reveals that even though the Rabbit processors can use a faster clock (up to 60MHz), the effective processing power is comparable to that of a similarly-priced AVR processor using a slower clock (up to 32MHz). For example, the "INC (IX+d)" instruction requires 12 clocks on a Rabbit processor. The equivalent instruction sequence (LDD, INC, STD) on an AVR requires 5 cycles. Another example is the *CALL* instruction. It requires 12 cycles on a Rabbit processor versus 4 to 5 cycles on an AVR processor. This difference, in part, is due to the AVR using on-chip memory for both instructions and data, whereas the Rabbit uses off-chip memory for both instructions and data.

For the study of Dynamic C, we used *Dynamic C Function Reference Manual*. Dynamic C is an integrated development system for writing embedded software. It is designed for use with Rabbit controllers and other controllers based on the Rabbit microprocessor. Perhaps the most notable feature of the Rabbit microcontroller is its development environment. Dynamic C, a product of Rabbit Semiconductor, has additions, deletions and inconsistencies compared to the ANSI-C standard. (Reference: Porting a Program to Dynamic C-Rabbit Semiconductor)

Dynamic C follows the ISO/ANSI C standard when feasible and desirable. Because the standard does not take into account the special needs of embedded systems, it is necessary to depart from the standard in some areas and desirable in others. The standard does not take into account important embedded systems issues such as read only memory and embedded assembly language. For this reason, practical compilers intended for embedded systems do not completely comply with the standard, but use it as a guide.

Rabbit Semiconductor claims that the instruction set of Rabbit processors is optimized for C code. A similar claim is made by Atmel for their AVR processors. The two architectures actually have very similar addressing modes, such as literal, register, indirect and indirect plus displacement. Furthermore, both architectures have specialized 16-bit registers. The Rabbit has the IX, IY and SP, whereas the AVR has X, Y and Z.

The main difference is that the Rabbit instructions place more constraints on register usage compared to the AVR instructions. For example, the 8-bit Rabbit *ADD* instruction permits only the A-register be the destination. However, the *ADD* instruction of the AVR permits the use any one of the 32 8-bit registers as the source or destination. Generally speaking, an instruction set that is less register restrictive is more optimizable because there is less need to save-and-reload the content of a register.

We also studied the software like Borland C; it creates one type of Graphical User Interface which is based on MS Dos. It is common platform for Embedded Systems; it works on most of the Embedded ICs.

The books viz. *Embedded Ethernet and Internet Complete Designing and Embedded Systems Design using the Rabbit 3000 Microprocessor interfacing, networking and application development* are used how to write the programs for TCP stack in the Rabbit Processors and also used to study sample programs of Rabbit Processor.

So by this way we have completed our study part and started working on the actual project and programming of Rabbit.

CHAPTER 3 - ETHERNET SIGNIFICANCE

3.1 TCP/IP LAYERS

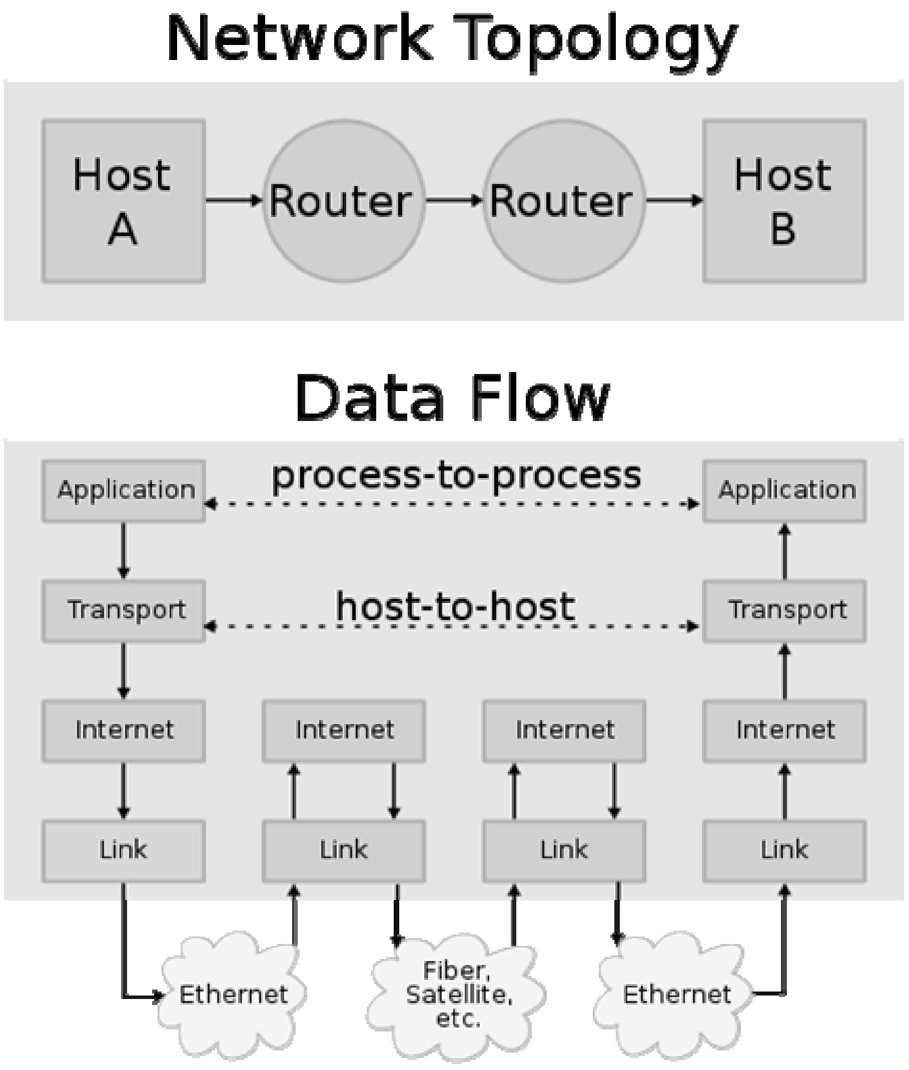


Figure 5 Illustration of TCP

The layers near the top are logically closer to the user application, while those near the bottom are logically closer to the physical transmission of the data. Viewing layers as providing or consuming a service is a method of abstraction to isolate upper layer protocols from the nitty-gritty detail of transmitting bits over, for example, Ethernet and collision detection, while the lower layers avoid having to know the details of each and every application and its protocol.

This abstraction also allows upper layers to provide services that the lower layers cannot, or choose not to, provide. Again, the original OSI model was extended to include connectionless services (OSIRM CL). For example, IP is not designed to be reliable and is a best effort delivery protocol. This means that all transport layer implementations must choose whether or not to provide reliability and to what degree. UDP provides data integrity (via a checksum) but does not guarantee delivery; TCP provides both data integrity and delivery guarantee (by retransmitting until the receiver acknowledges the reception of the packet).

This model lacks the formalism of the OSI model and associated documents, but the IETF does not use a formal model and does not consider this a limitation, as in the comment by David D. Clark, "We reject: kings, presidents and voting. We believe in: rough consensus and running code." Criticisms of this model, which have been made with respect to the OSI model, often do not consider ISO's later extensions to that model.

1. For multiaccess links with their own addressing systems (e.g. Ethernet) an address mapping protocol is needed. Such protocols can be considered to be below IP but above the existing link system. While the IETF does not use the terminology, this is a subnetwork dependent convergence facility according to an extension to the OSI model, the internal organization of the network layer (IONL).
2. ICMP & IGMP operate on top of IP but do not transport data like UDP or TCP. Again, this functionality exists as layer management extensions to the OSI model, in its *Management Framework*(OSIRM MF)
3. The SSL/TLS library operates above the transport layer (uses TCP) but below application protocols. Again, there was no intention, on the part of the designers of these protocols, to comply with OSI architecture.
4. The link is treated like a black box here. This is fine for discussing IP (since the whole point of IP is it will run over virtually anything). The IETF explicitly does not intend to discuss transmission systems, which is a less academic but practical alternative to the OSI model.

The following is a description of each layer in the TCP/IP networking model starting from the lowest level.

3.1.1 Link layer-

The link layer is the networking scope of the local network connection to which a host is attached. This regime is called the *link* in Internet literature. This is the lowest component layer of the Internet protocols, as TCP/IP is designed to be hardware independent. As a result TCP/IP is able to be implemented on top of virtually any hardware networking technology.

The link layer is used to move packets between the Internet layer interfaces of two different hosts on the same link. The processes of transmitting and receiving packets on a given link can be controlled both in the software device driver for the network card, as well as on firmware or specialized chipsets. These will perform data link functions such as adding a packet header to prepare it for transmission, and then actually transmit the frame over a physical medium. The TCP/IP model includes specifications of translating the network addressing methods used in the Internet Protocol to data link addressing, such as Media Access Control (MAC), however all other aspects below that level are implicitly assumed to exist in the link layer, but are not explicitly defined.

This is also the layer where packets may be selected to be sent over a virtual private network or other networking tunnel. In this scenario, the link layer data may be considered application data which traverses another instantiation of the IP stack for transmission or reception over another IP connection. Such a connection, or virtual link, may be established with a transport protocol or even an application scope protocol that serves as a tunnel in the link layer of the protocol stack. Thus, the TCP/IP model does not dictate a strict hierarchical encapsulation sequence.

3.1.2 Internet layer-

The internet layer has the responsibility of sending packets across potentially multiple networks. Internetworking requires sending data from the source network to the destination network. This process is called routing.

In the Internet protocol suite, the Internet Protocol performs two basic functions:

- *Host addressing and identification*: This is accomplished with a hierarchical addressing system (see IP address).
- *Packet routing*: This is the basic task of sending packets of data (datagrams) from source to destination by sending them to the next network node (router) closer to the final destination.

IP can carry data for a variety of different upper layer protocols. These protocols are each identified by a unique protocol number: for example,Internet Control Message Protocol (ICMP) and Internet Group Management Protocol (IGMP) are protocols 1 and 2, respectively.

Some of the protocols carried by IP, such as ICMP (used to transmit diagnostic information about IP transmission) and IGMP (used to manage IP Multicast data) are layered on top of IP but perform internetworking functions. This illustrates the differences in the architecture of the TCP/IP stack of the Internet and the OSI model.

3.1.3 Transport layer-

The responsibility of the transport layer includes end-to-end message transfer independent of the underlying network, along with error control, segmentation, flow control, congestion control, and application addressing (port numbers). End to end message transmission or connecting applications at the transport layer can be categorized as connection-oriented, implemented in TCP, or connectionless, implemented in UDP.

The transport layer can be thought of as a transport mechanism, e.g., a vehicle with the responsibility to make sure that its contents (passengers/goods) reach their destination safely and soundly, unless another protocol layer is responsible for safe delivery. The transport layer provides this service of connecting applications through the use of service ports. Since IP provides only a best effort delivery, the transport layer is the first layer of the TCP/IP stack to offer reliability. IP can run over a reliable data link protocol such as the High-Level Data Link Control (HDLC). Protocols above transport, such as RPC, also can provide reliability.

For example, the TCP is a connection-oriented protocol that addresses numerous reliability issues to provide a reliable byte stream:

- data arrives in-order
- data has minimal error (i.e. correctness)
- duplicate data is discarded
- lost/discarded packets are resent
- includes traffic congestion control

The newer Stream Control Transmission Protocol (SCTP) is also a reliable, connection-oriented transport mechanism. It is message-stream-oriented — not byte-stream-oriented like TCP — and provides multiple streams multiplexed over a single connection. It also provides multi-homing support, in which a connection end can be represented by multiple IP addresses (representing multiple physical interfaces), such that if one fails, the connection is not interrupted. It was developed initially for telephony applications (to transport SS7 over IP), but can also be used for other applications.

User Datagram Protocol is a connectionless datagram protocol. Like IP, it is a best effort, "unreliable" protocol. Reliability is addressed through error detection using a weak checksum algorithm. UDP is typically used for applications such as streaming media (audio, video,Voice over IP etc.) where on-time arrival is more important than reliability, or for simple query/response applications like DNS lookups, where the overhead of setting up a reliable connection is disproportionately large. Real-time Transport Protocol (RTP) is a datagram protocol that is designed for real-time data such as streaming audio and video.

TCP and UDP are used to carry an assortment of higher-level applications. The appropriate transport protocol is chosen based on the higher-layer protocol application. For example, the File Transfer Protocol expects a reliable connection, but the Network File System (NFS) assumes that the subordinate Remote Procedure Call protocol, not transport, will guarantee reliable transfer. Other applications, such as VoIP, can tolerate some loss of packets, but not the reordering or delay that could be caused by retransmission. The applications at any given network address are distinguished by their TCP or UDP port. By convention certain *well known ports* are associated with specific applications.

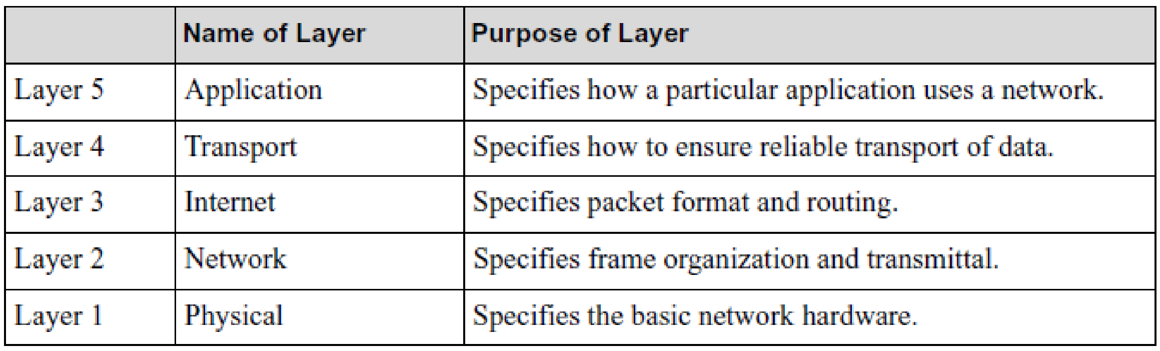
3.1.4 Application layer-

The application layer contains the higher-level protocols used by most applications for network communication. Examples of application layer protocols include the File Transfer Protocol (FTP) and the Simple Mail Transfer Protocol (SMTP). Data coded according to application layer protocols are then encapsulated into one or (occasionally) more transport layer protocols (such as TCP or UDP), which in turn use lower layer protocols to effect actual data transfer.

Since the IP stack defines no layers between the application and transport layers, the application layer must include any protocols that act like the OSI's presentation and session layer protocols. This is usually done through libraries.

Application layer protocols generally treat the transport layer (and lower) protocols as black boxes which provide a stable network connection across which to communicate, although the applications are usually aware of key qualities of the transport layer connection such as the end point IP addresses and port numbers. As noted above, layers are not necessarily clearly defined in the Internet protocol suite. Application layer protocols are most often associated with client–server applications, and the commoner servers have specific ports assigned to them by the IANA: HTTP has port 80; Telnet has port 23; etc. Clients, on the other hand, tend to use ephemeral ports, i.e. port numbers assigned at random from a range set aside for the purpose.

Transport and lower level layers are largely unconcerned with the specifics of application layer protocols. Routers and switches do not typically "look inside" the encapsulated traffic to see what kind of application protocol it represents; rather they just provide a conduit for it. However, some firewall and bandwidth throttling applications do try to determine what's inside, as with the Resource Reservation Protocol (RSVP). It's also sometimes necessary for Network Address Translation (NAT) facilities to take account of the needs of particular application layer protocols. (NAT allows hosts on private networks to communicate with the outside world via a single visible IP address using port forwarding, and is an almost ubiquitous feature of modern domestic broadband routers).



	Name of Layer	Purpose of Layer
Layer 5	Application	Specifies how a particular application uses a network.
Layer 4	Transport	Specifies how to ensure reliable transport of data.
Layer 3	Internet	Specifies packet format and routing.
Layer 2	Network	Specifies frame organization and transmittal.
Layer 1	Physical	Specifies the basic network hardware.

Figure 6 TCP layer & functions

3.2 Ethernet Frame format-

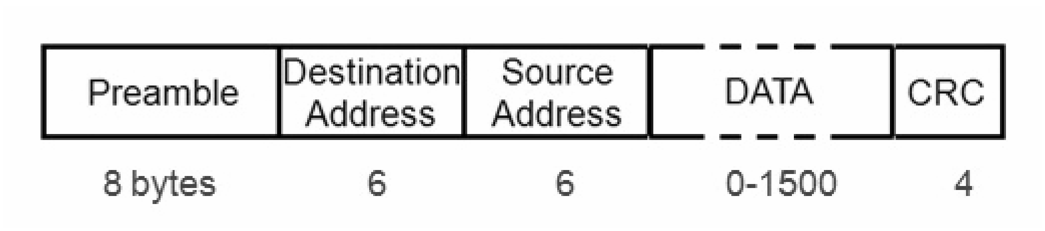


Figure 7 Ethernet Frame format old

TCP/IP (Transmission Control Protocol/Internet Protocol) is a set of protocols independent of the physical medium used to transmit data, but most data transmission for Internet communication begins and ends with Ethernet frames.

The Ethernet can use either a bus or star topology. A bus topology attaches all devices in sequence on a single cable. In a star topology all devices are wired directly to a central hub. 10Base-T uses a combination called a star-shaped bus topology because while the attached devices can share all data coming in on the cable, the actual wiring is in a star shape.

The access method used by the Ethernet is called Carrier Sense Multiple Access with Collision Detect (CSMA/CD). This is a contention protocol, meaning it is a set of rules to follow when there is competition for shared resources.

3.3 Ethernet Address-

All Ethernet interfaces have a unique 48-bit address that is supplied by the manufacturer. It is called the Ethernet address (also known as the MAC address, for Media Access Control). Ethernet-enabled Rabbit boards store this value in Flash Memory (EEPROM) that is programmed at the factory. If you need unique Ethernet addresses for some product you are making, you can obtain them from the IEEE Registration Authority.

3.4 Physical Connections-

A Realtek RTL8019 10Base-T interface chip provides a 10 Mbps Ethernet connection. This chip is used on many Ethernet-enabled Rabbit boards. The corresponding port can be connected directly to an Ethernet network.

By using hubs and routers, a network can include a large number of computers. A network might include all the computers in a particular building. A local network can be connected to the Internet by means of a gateway. The gateway is a computer that is connected both to the local net work and to the Internet. Data that must be sent out over the Internet are sent to the local network interface of the gateway, and then the gateway sends them on to the Internet for routing to some other computer in the world. Data coming in from the Internet are directed to the gateway, which then sends them to the correct recipient on the local network.

3.5 Cables-

Ethernet cables are similar to U.S. telephone plug cables, except they have eight connectors. For our purposes, there are two types of cables—crossover and straight-through. In most instances, the straight-through cables are used. It is necessary to use a crossover cable when two computers are connected directly without a hub (for example, if you want to connect your PC's Ethernet directly to the Rabbit Semiconductor TCP/IP Development Board.) Some hubs have one input that can accept either a straight-through or crossover cable depending on the position of a switch. In this case make sure that the switch position and cable type agree.

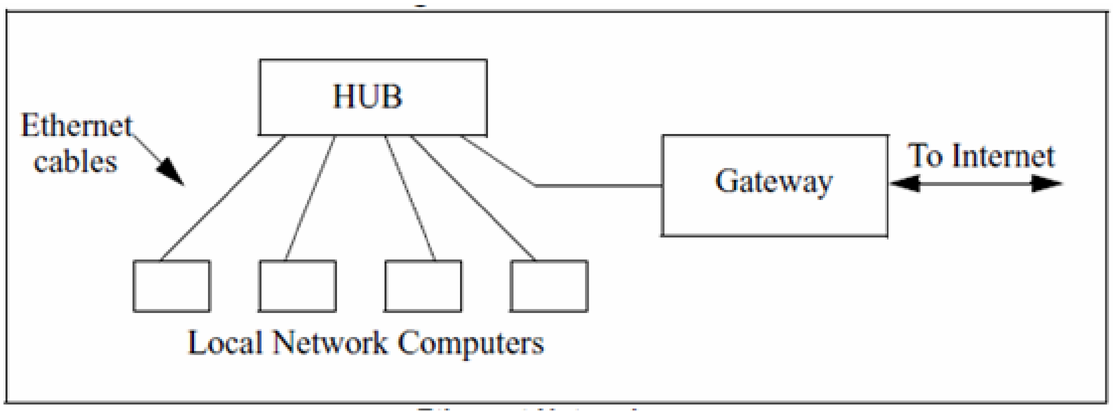


Figure 8 Ethernet Network

3.6 Frames Updated-

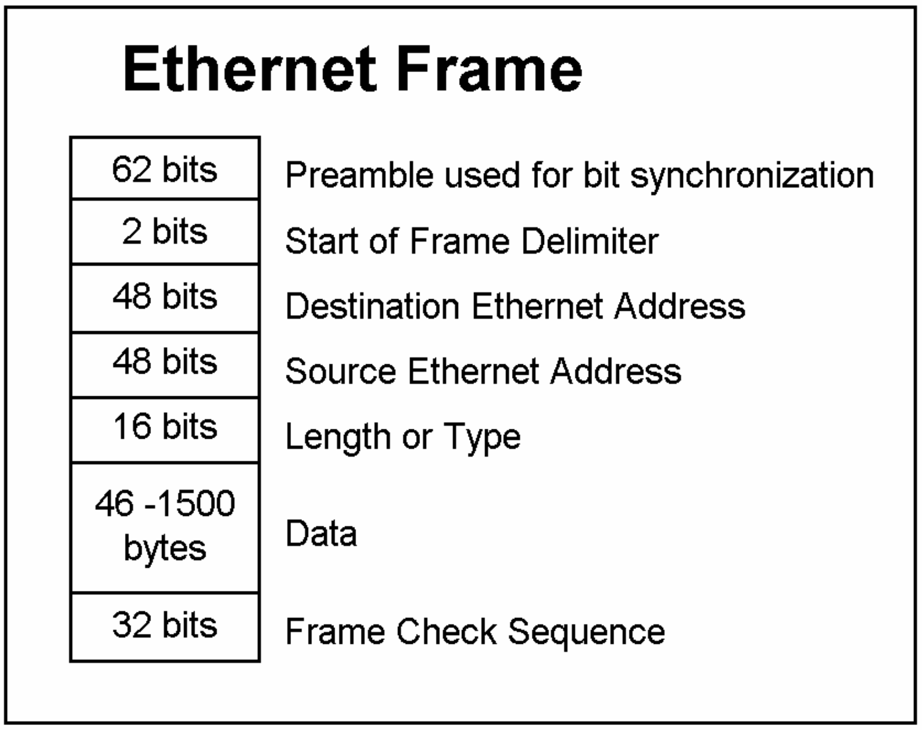


Figure 9 Ethernet Frame format updated

Bits flowing across the Ethernet are grouped into structures called frames. A frame must be between 46 and 1500 bytes in size. An Ethernet frame has four parts:

- ✓ A Preamble of 8 bytes that helps synchronize the circuitry, thus allowing small bit rate differences between sender and receiver.
- ✓ A Header of 14 bytes that contains a 6 byte destination address, 6 byte source address and a 2 byte type field.
- ✓ A Data area of variable length that, along with the header, is passed to the IP layer (the Network layer).
- ✓ A Trailer of 4 bytes that contains a CRC to guard against corrupted frames.

If the destination address is all 1 bits, it defines a broadcast frame and all systems on the local network process the frame. There are also multicast frames. A subset of systems can form a “multicast” group that has an address that does not match any other system on the network. All systems in a particular subset process a packet with a destination address that matches their subset. A system can belong to any number of subsets.

A system may put its interface(s) into promiscuous mode and process all frames sent across its Ethernet. This is known as "sniffing the ether." It is used for network debugging and spying.

3.7 Collisions-

In a star-shaped bus topology, all systems have access to the network at any time. Before sending data, a system must determine if the network is free or if it is already sending a frame. If a frame is already being sent, a system will wait. Two systems can “listen” on the network and “hear” silence and then proceed to send data at the same time. This is called a collision. Ethernet hardware has collision detection sensors to take care of this problem. This is the Collision Detect (CD) part of CSMA/CD. The colliding data is ignored, and the systems involved will wait a random amount of time before resending their data.

3.8 TCP/IP Protocol Stack-

TCP/IP is the protocol suite upon which all Internet communication is based. Different vendors have developed other networking protocols, but even most network operating systems with their own protocols, such as Netware, support TCP/IP. It has become the de facto standard.

Protocols are sometimes referred to as protocol stacks or protocol suites. A protocol stack is an appropriate term because it indicates the layered approach used to design the networking software.

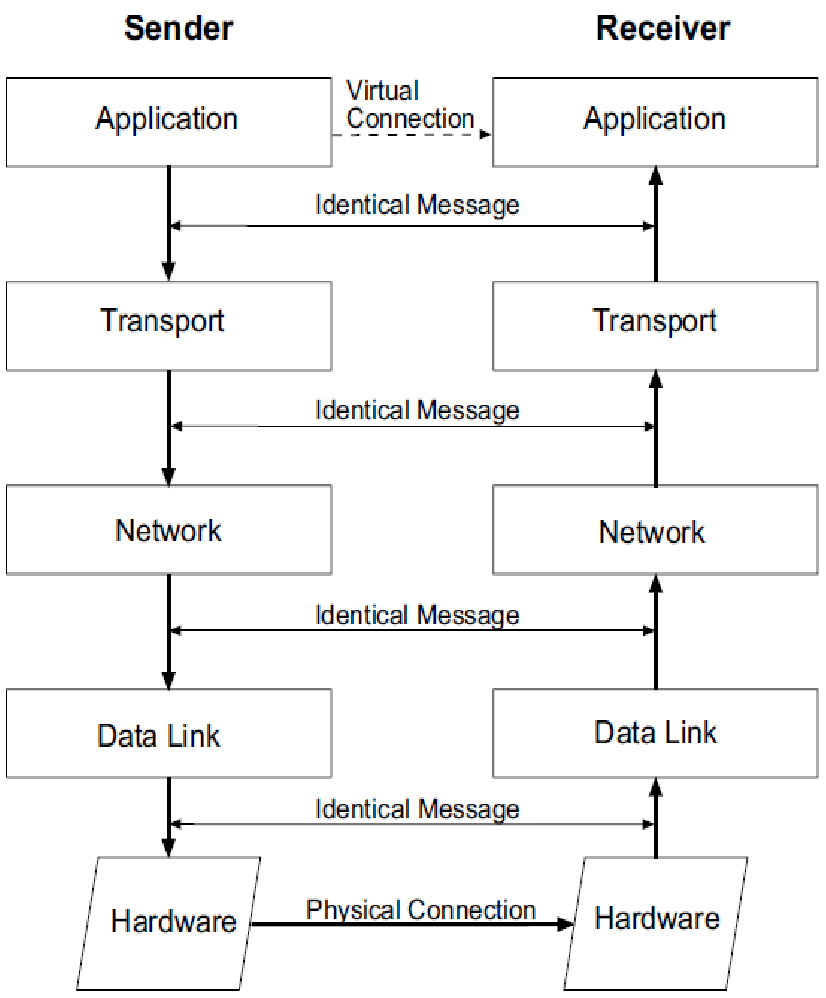


Figure 10 Flow of Data Between Two Computers Using TCP/IP Stacks

Each host or router in the internet must run a protocol stack. The details of the underlying physical connections are hidden by the software. The sending software at each layer communicates with the corresponding layer at the receiving side through information stored in headers. Each layer adds its header to the front of the message from the next higher layer. The header is removed by the corresponding layer on the receiving side.

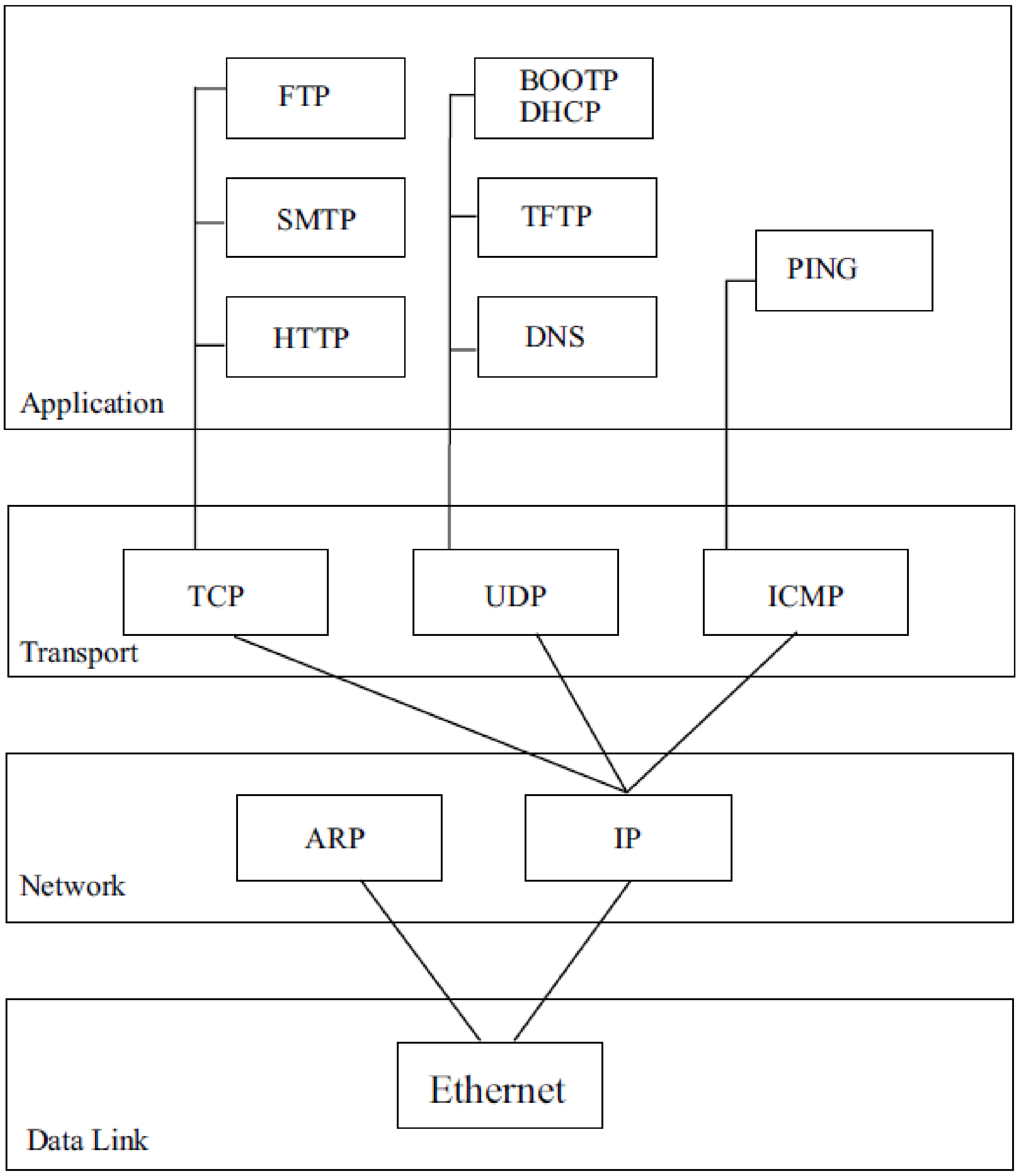


Figure 11 TCP/IP Protocol Flow

3.9 Comparison RS232, RS485 and TCP/IP-

RS232: One master and one slave. Typically a cable with 3 conductors with max length of approx a couple of hundred feet. Usually easy. Sometimes some jumpers are required at one end to defeat handshaking. RS-232 is point-to-point only and short distance, typically 20m. Not generally useful for 'multiple dispensers' without multiple serial ports and associated protocol and application software.
RS485: One master and up to 128 slaves but take care to read more if you plan on more than 32. There are two wiring systems – so called 2-wore and so called 4-wire. They can be incompatible but usually 4-wire devices can be made to work on 2-wire systems. Each device must have a unique address and all devices must be set to the same baud rate, data bits, stop bits and parity. Usually easy to implement. The RS485 physical layer allows up to 128 devices to be installed on a single network with a max physical length of 4000ft and speeds up to 115k baud. Using repeaters allows the length to be increased. Compare to Ethernet where the spec allows a max of 100 meters (330ft) on a single unrepeated segment.

Distances to 1000m, rates dependent on distance, can range from the low hundreds/Kb at long distance to 1.5Mb at short distances, daisy chain/multidrop topology, balanced differential line but subject to common mode and bias issues.
TCP/IP: All devices are essentially peers. A single device can be a master and a server. Routers can be used to connect sub-nets together. Broadcasts are almost ever used so are not an issue. 100m distance, rates to 100Mb or 1 GB, hub and spoke topology, transformer coupled signals minimizes common mode issues.

The noticeable differences are-

RS-485 can have a "bus" type multi-drop connection (multiple connections to the same cable), while modern Ethernet uses a "hub" type connection. While that sounds like an advantage for RS-485, a "hub and spoke" installation is often a lot easier in practice when you look at where you will run the cables. You can typically run a single Ethernet cable out to a switch located at a cluster of nodes, and then run another cable from that switch out to another switch at another cluster of nodes, etc. There's a limit to how many "hops" you can do that for, but whether that is a problem in practice depends on your physical layout.

RS-485 wiring and termination problems are a routine question that we get here, so the problems with RS-485 are probably quite common. Ethernet seems to be less prone to installation problems. However, if you are not using off the shelf Ethernet cables, make sure the wiring contractor connects the right wires to the right pins! They can sometimes hook it up wrong and things will still mostly work. The problems usually relate to mixing and matching wires from different pairs instead of keeping the pairs together.

"Serial servers" or "terminal servers" are a common way of connecting serial devices over Ethernet, so you can often use serial devices with Ethernet. If you are running you're wiring a long distance outdoors, then you might want to look at fiber optic cable, as that may give you better resistance to lightning.

If I was putting in a new installation today and I had a choice, I would prefer Ethernet and TCP/IP over RS-485. I suspect that RS-485 will eventually disappear from the market, while I expect Ethernet to be around for a very long time to come.

CHAPTER 4 - SYSTEM DESCRIPTION

4.1 Design Advantages-

- ✓ Designed for embedded networking with intelligence and I/O control
- ✓ Serves web pages, controls remote devices, links equipment to the Internet Security-key feature with encryption capabilities adds peace of mind for OEMs and systems integrators
- ✓ Complete microprocessor, on-board memory, development software with royalty-free TCP/IP stack, and hundreds of sample programs reduce time-to-market

Rabbit Semiconductor provides low-cost, quick-to-market solutions for a variety of networked embedded systems. Rabbit's RabbitCore microprocessor core modules and customizable single-bard computer solutions offer fast and powerful development solutions with a variety of form factors, I/O configurations, and memory. These solutions are used in building security, point of sale, parking systems, telecommunications, vehicle and ship systems, container tracking and a broad variety of similar applications. Rabbit bundles hardware and software together, creating an engineer-friendly development environment.

- ✓ Glueless architecture of Rabbit 4000

Glue-less means that no other silicon is necessary to connect the processors together. For example, The Nehalem EX with four full QPI can support glue-less 8-way system architecture. Each processor connects directly to all seven other processors with a half-wide QPI link, and uses the remaining half wide QPI to connect to the IOH.

The "glue" people refer to are those chips with a few gates in them. Glue-less means that there is no need to invert or buffer or whatever the odd signal here and there. All has been thought out. These chips are also called jelly bean chips sometimes. Getting rid of them is good for many reasons, including reduced cost in volume applications.

- ✓ Generates low EMI of just 25dBµV

As microprocessor speeds increase, reducing Electromagnetic Interference (EMI) becomes an essential part of design considerations. This application note provides guidelines to aid the PC desktop system and motherboard designer with low cost solutions on reducing EMI. The document focuses on the efforts made by Rabbit Semiconductors to prevent systems utilizing Rabbit Semiconductors processors and components from interfering with other electronic products. There are generally two methods by which interference is measured:

• below 30 MHz EMI RF, noise is measured as *conducted* emissions

• above 30 MHz RF, noise is measured as *radiated* emissions.

As all of the frequencies generated for and by the P6 class of microprocessor (e.g., Pentium® III processor, Pentium® II processor, Rabbit Semiconductors® 4000, and Pentium® Pro processor) exceed 50 MHz, this document concerns itself only with radiated emissions above 50 MHz

- ✓ Supports 10BaseT Ethernet
- ✓ Memory Organization and Compiling Program

In time of compiling the Program which one has written at Dynamic C, one has a choice of doing his or her software development in the flash memory or in the Static RAM included on the RCM4000. The Flash Memory and SRAM options are selected with the OPTIONS > PROGRAM OPTION > COMPILER menu. The Advantage of working in RAM is to save wear the flash memory, which is limited to about 100,000 write cycles. The Disadvantage is that the code and data might not both fit in RAM. An operation can be compiled in RAM, but cannot run standalone from RAM after the programming cable is disconnected. All standalone applications can run only from flash memory.

- ✓ Standard Debugging Features

There are lots of standard debugging features in Dynamic C which help user to make understood what is happening in time of running of any programming. Those features are:

1. Breakpoints: Set breakpoints that can disable interrupts.

2. Single-Stepping: Step into or over functions at a source or machine code level µCOS-II aware. One can enter into the Single-stepping mode after compiling the program successfully by pressing the key <F8>.
3. Code disassembly: The disassembly window displays addresses, opcodes, and machine cycle times. Switch between debugging at machine-code level and source-code level by simply opening or closing the disassembly window.
4. Watch Expression: Watch expressions are compiled when defined, so complex expressions including function calls may be placed into watch expressions. Watch expressions can be updated with or without stopping program execution.
5. STDIO Window: printf outputs to this window and keyboard input on host PC can be detected for debugging purposes. Printf outputs may also be sent to serial port or file.

4.2 Disadvantages-

- ✓ More costly than other processors like ARM, Intel, etc.
- ✓ Less availability of Experts in the Rabbit market

4.3 Applications-

- ✓ Serial-to-Ethernet applications
- ✓ Remote monitoring and communications
- ✓ Web-enabling devices
- ✓ Device/data management and control

CHAPTER 5 - HARDWARE DESCRIPTION

5.1 Rabbit 4000 Module-

The Rabbit 4000 has several powerful design features that practically eliminate EMI problems, which is essential for OEMs that need to pass CE and regulatory radiofrequency emissions tests. The amplitude of any electromagnetic radiation is reduced by the internal spectrum spreader, by gated clocks (which prevent unnecessary clocking of unused registers), and by separate power planes for the processor core and I/O pins (which reduce noise crosstalk). An auxiliary I/O bus can be used by designers to enable separate buses for I/O and memory or to limit loading the memory bus to reduce EMI and ground bounce problems when interfacing external peripherals to the processor. The auxiliary I/O bus accomplishes this by duplicating the Rabbit's data bus on Parallel Port A, and uses Parallel Port B to provide the processor's six or eight least significant address lines for interfacing with external peripherals.

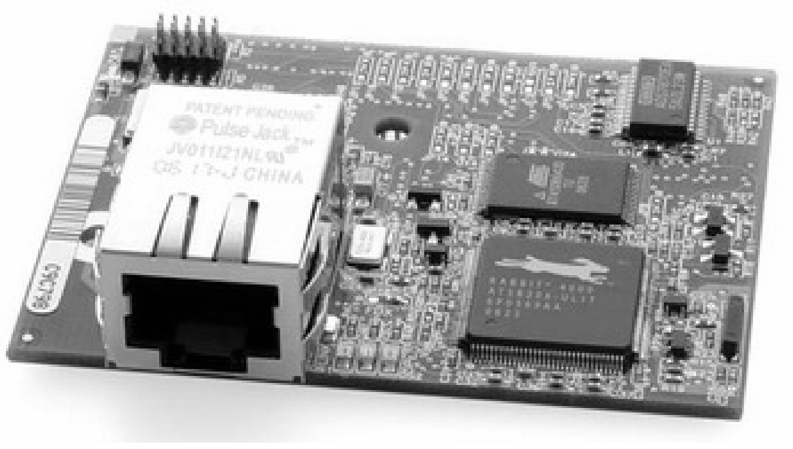


Figure 12 Rabbit 4000 Module with 4300 Processor

The high-performance instruction set offers both greater efficiency and execution speed of compiler-generated C code. Instructions include numerous single-byte opcodes that execute in two clock cycles, 16-bit and 32-bit loads and stores, 16-bit and 32-bit logical and arithmetic operations, 16 × 16 multiply (executes in 12 clocks), long jumps and returns for accessing a full 16 megabytes of memory, and one-byte prefixes to turn memory-access instructions into internal and external I/O instructions. Hardware-supported breakpoints ease debugging by trapping on code execution or data reads and writes.

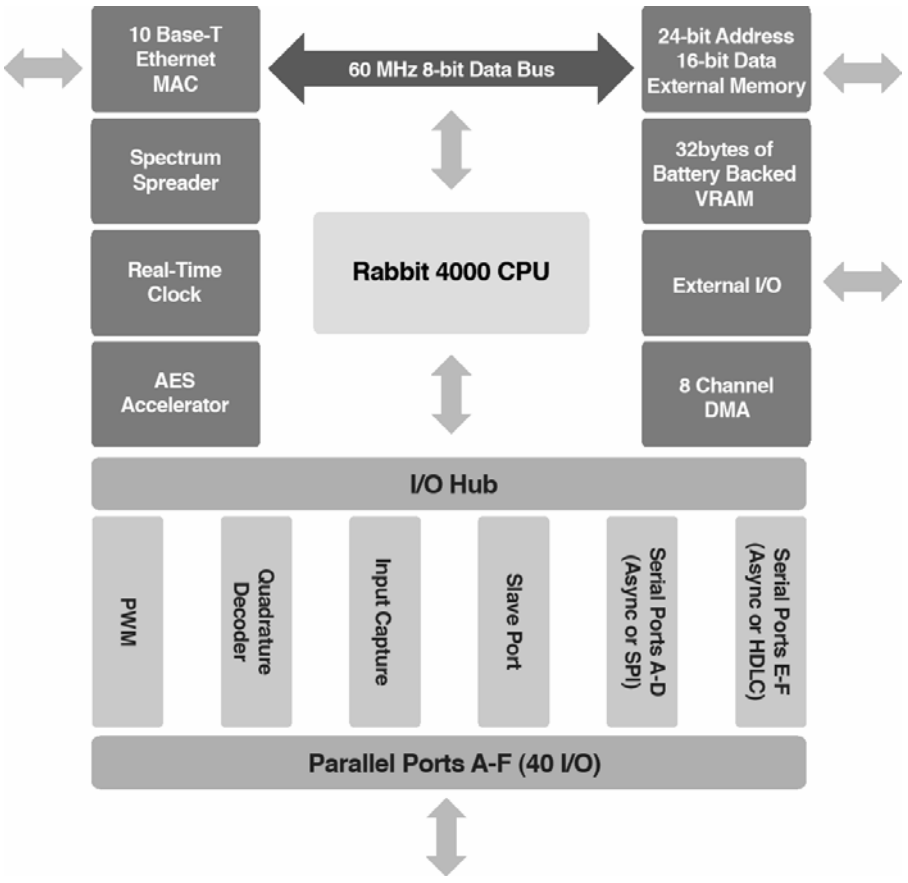


Figure 13 Rabbit 4000 CPU Block Diagram

Development Kit comes complete with:

- ✓ RCM4300 RabbitCore Module
- ✓ Prototyping Board
- ✓ Serial cable for programming and debugging
- ✓ Dynamic CÂ® integrated development software
- ✓ Getting Started Instructions
- ✓ Complete product documentation on CD including the Rabbit 4000 reference manual
- ✓ 240VAC adapter
- ✓ Rabbit 4000 pin specifications poster

- ✓ Connectors and accessories

5.1.1 CLOCKS:

The Rabbit 4000 supports up to three separate clocks— the main clock, the 32 kHz clock, and the 20 MHz Ethernet clock.

The Main Clock is used to derive the processor clock and the peripheral clock inside the processor.

The 32 KHz Clock is used to drive the asynchronous serial bootstrap, the real-time clock, the periodic interrupt, and the watchdog timers. If these features are not used in a design, the use of the 32 kHz clock is optional.

The Ethernet Clock can be driven by the processor clock, the processor clock divided by 2, or by the input on PE6. The Ethernet clock needs to be 20 MHz to conform to the 10Base-T specification.

The Rabbit 4000 has a spectrum spreader on the main clock that shortens and lengthens clock cycles. This has the net effect of reducing the peak energy of clock harmonics by spreading the spectral energy into nearby frequencies, which reduces EMI and facilitates government-mandated EMI testing.

The *Various Registers* which are being used here are:

Global Control/Status Register (GCSR), Global Clock Modulator 0 Register (GCM0R), Global Clock Modulator 1 Register (GCM1R), Global Clock Double Register (GCDR), Global Output Control Register (GOCR), etc.

5.1.2 Rabbit System Management:

There are a number of basic system peripherals in the Rabbit 4000 processor. The peripherals which will be covered briefly in this part are the *periodic interrupt, the real-time clock, the watchdog timers, the battery-backed onchip-encryption RAM,* and some of the miscellaneous output pins and their control and processor registers that provide the processor ID and revision numbers.

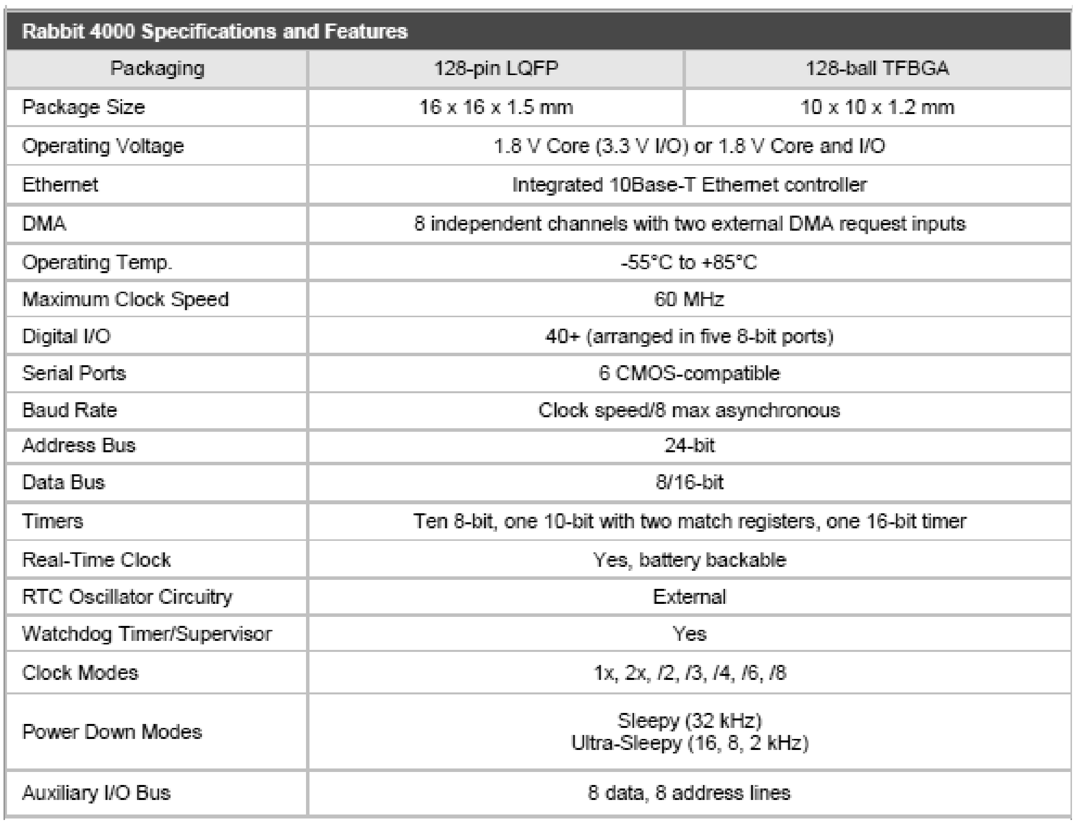
The Periodic Interrupt, when enabled, is generated every 16 clocks of the 32 kHz clock (every 488 µs, or 2.048 kHz). This interrupt can be used to perform periodic tasks.

The Real-Time Clock (RTC) consists of a 48-bit counter that is clocked by the 32 kHz clock. It is powered by the VBAT pin, and so can be battery-backed.

The Watchdog Timers are clocked by the 32 kHz clock. There are two Watchdog Timers in Rabbit 4000 Microprocessor Chip. The main watchdog timer can be set to time out from 250 ms to 2 seconds, while the secondary watchdog timer can time out from 30.5 µs up to 7.8 ms. The Main Watchdog Timer can reset the processor if not reload within the time, while the secondary watchdog timer generates a Priority 3 secondary watchdog interrupt when it is not reset within the time.

The battery-backed onchip-encryption RAM consists of 32 bytes of memory that are powered by the VBAT pin. Their values are not affected by reset, but are erased if the state of the SMODE pins changes.

5.1.3 Rabbit 4000 Specifications-



Rabbit 4000 Specifications and Features		
Packaging	128-pin LQFP	128-ball TFBGA
Package Size	16 x 16 x 1.5 mm	10 x 10 x 1.2 mm
Operating Voltage	1.8 V Core (3.3 V I/O) or 1.8 V Core and I/O	
Ethernet	Integrated 10Base-T Ethernet controller	
DMA	8 independent channels with two external DMA request inputs	
Operating Temp.	-55°C to +85°C	
Maximum Clock Speed	60 MHz	
Digital I/O	40+ (arranged in five 8-bit ports)	
Serial Ports	6 CMOS-compatible	
Baud Rate	Clock speed/8 max asynchronous	
Address Bus	24-bit	
Data Bus	8/16-bit	
Timers	Ten 8-bit, one 10-bit with two match registers, one 16-bit timer	
Real-Time Clock	Yes, battery backable	
RTC Oscillator Circuitry	External	
Watchdog Timer/Supervisor	Yes	
Clock Modes	1x, 2x, /2, /3, /4, /6, /8	
Power Down Modes	Sleepy (32 kHz) Ultra-Sleepy (16, 8, 2 kHz)	
Auxiliary I/O Bus	8 data, 8 address lines	

Figure 14 Rabbit 4000 Specifications

5.1.4 Memory Management:

The Rabbit 4000 supports both 8-bit and 16-bit external flash and SRAM devices; three chip selects and two read/write-enable strobes allow up to six external devices to be attached at once. The 8-bit mode allows 0, 1, 2, or 4 wait states to be specified for each device, and the 16-bit mode allows 0 to 9 wait states depending on the settings. Both 8-bit and 16-bit page-mode devices are also supported.

The Rabbit 4000's physical memory space contains four consecutive banks, each of which can be mapped to an individual chip-select/enable strobe pair. The banks can be set for equal sizes ranging from 128KB up to 4MB, providing a total physical memory range from 512KB up to 16MB.

The *Block Diagram* for Memory Management is shown below:

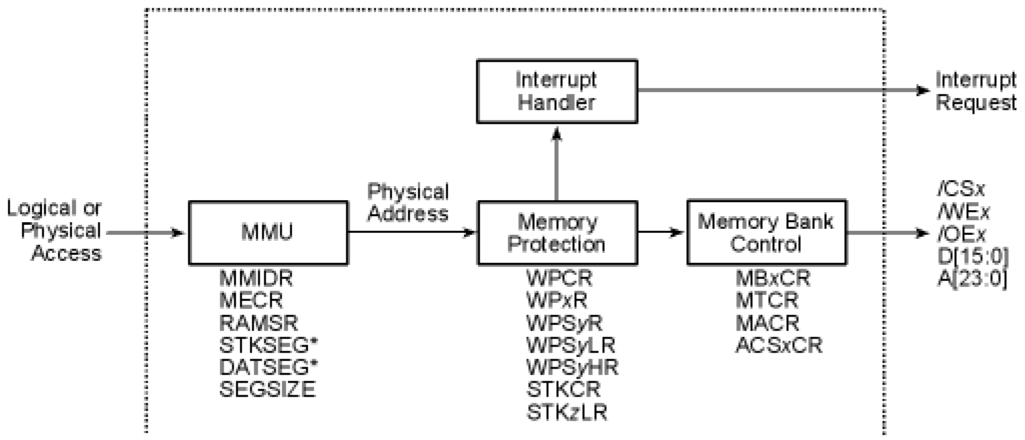


Figure 15 Memory Management in Rabbit Processor

The *Various Registers* which are being used here are described below:

MMU Instruction/Data Register (MMIDR) *(Address = 0x0010)* is being used for mainly select the Data Line and Address Line by selecting the higher eight bits of the internal I/O address bus.

Stack Segment Register (STKSEG*) (Address = 0x0011)* is being used to read the current contents of this register or to write eight LSB-s of physical address offset to use if SEGSIZ[7:4] ≤ Addr[15:12] < 0xE.

Stack Segment Low Register (STKSEGL*) (Address = 0x001A)* is doing the same operation like previous. The difference is that STKSEGL is doing the operation on the four LSB-s, while the previous one is doing the operation on eight LSB-s.

Some other important registers are *Stack Segment High Register (STKSEGH), Data* **Segment Register (DATSEG),** etc. There are more Registers to control the operation in Memory Management. If any one is interested about those things please consult the Users' Manual of Rabbit4000 Microprocessor in the site of Rabbit Semiconductor.

5.1.5 Various parallel Ports:

There are five parallel Ports in Rabbit4000 Microprocessor Chips. Those Parallel Ports are: Parallel Port A, Parallel Port B, Parallel Port C, and Parallel Port D, Parallel Port E. These parallel ports have some registers to handle those ports for various purposes (to blink LEC, to display some messages by LCD Display, etc.) by writing the perfect Software in the environment of Dynamic C. Some of the parallel ports have some dual characteristics i.e. beside their own job they can be used for other purposes also. Each Parallel port has 8 pins. A detailed and in-depth description about those five parallel Ports will be held later while describing our main project since in our main project we have used those parallel Ports in a wide range.

5.1.6 Various Timers:

There are four types of Timers in Rabbit4000 Microprocessor. They are Timer A, Timer B and Timer C. The three Timers also used various Registers. But we are not going into the details of those registers since for our Project it is not needed.

TIMER A:

The Timer A peripheral consists of ten separate eight-bit countdown timers, A1–A10. Each counter counts down from a programmed time constant, which is automatically reloaded into the respective counter when the count reaches zero.

TIMER B:

The Timer B peripheral consists of a ten-bit free running up-counter, two match registers, and two step registers. Timer B is driven by perclk/2, by per clk/16, or by the output of timer A1.

TIMER C:

The Timer C peripheral is a 16-bit up-counter clocked by the peripheral clock divided by 2, by the peripheral clock divided by 16, or by the output of countdown timer A1.

5.1.7 Various Serial Ports:

There are 6 Serial Ports in the Rabbit 4000 Microprocessor Chip. The six Serial Ports are; Serial Port A, Serial Port B, Serial Port C, Serial Port D, Serial Port E, Serial Port F. Among those six serial Ports Serial Port A, Serial Port B, Serial Port C and Serial Port D are identical, except for the source of data clock and the transmit, receive, and clock pins. The rest two Serial Ports i.e. Serial port E and Serial Port F are identical to each other, and their asynchronous operation is identical to that of Serial Ports A-D except for the source of the Data clock, the buffer sizes, and the transmit, receive, and the clock pins.

5.1.8 Slave Port:

The Slave Port is a parallel communication port that can be used to communicate with an external master device. The Slave port consists of three data input and data output registers, and a Status Register. The data Input registers are written by the master i.e. the external devices and are read by the processor. The data output registers are written by the processor and read by the master. Parallel Port A can be used as a Slave port. Since we have told in the earlier that Parallel Ports have some dual characteristics, so here among those five parallel ports Parallel Port A can be programmed as a Slave port in the case of Master-Slave communication between two Prototype Boards. Since in our Project the Slave Port is not used that much so we are not going in the detailed description about the Slave Port.

5.1.9 DMA Channels:

There are eight independent DMA channels on the Rabbit 4000. All eight channels are identical, and are capable of transferring data to or from memory, external I/O, or internal I/O. The priority between the channels can be either fixed or rotating, and the DMA use of the bus can be limited to guarantee interrupt latency or CPU throughput. The DMA channels are capable of special handling for the last byte of data when sending data to selected internal I/O addresses (such as the HDLC serial ports or to the Ethernet peripheral), and can also transfer end-of-frame status after transferring data from selected internal I/O addresses. The DMA channels are inherently byte-oriented. There are also some registers like *DMA Master Control/Status Register*, *DMA master Auto-Load Register*, *DMA Master Halt Register*, etc.

The above description is the basic things that how DMA channels work. Here also we are not going in the depth of this channels since in our project we have hardly used those channels.

5.1.10 Pulse Width Modulation:

The Pulse Width Modulator (PWM) consists of a 10-bit free running counter and four width registers. A PWM output consists of a train of periodic pulses within a 1024-count frame with a duty cycle that varies from 1/1024 to 1024/1024. Each PWM output is high for (n + 1) counts out of the 1024-clock count cycle, where n is the value held in the width register. The PWM is clocked by the output of Timer A9 which is used to set the period. Each PWM output high time can optionally be spread throughout the cycle to reduce ripple on the externally filtered PWM output. The PWM outputs can be passed through a filter and used as a 10-bit D/A converter. The outputs can also be used to directly drive devices such as motors or solenoids that have intrinsic filtering. That is the basic description of how PWM works in Rabbit4000 Microprocessor Chip. We are skipping from it since we did not use that PWM in our project.

5.1.11 10BASE-T Ethernet:

That is the most important feature of Rabbit4000 Microprocessor. But in our Project we didn't use this inbuilt Ethernet Facility. It is designed for using with Rabbit controllers and other controllers based on the Rabbit microprocessor chip. But after replacing the MCM card by Rabbit Card totally, GMRT is going to use this most important facility for sure. Now i am describing it in very short.

Network Port A implements all of the required digital elements of the 10Base-T standard, and is normally used with two channels of the DMA controller. The receiver provides 32 bytes of buffering, and the transmitter has 16 bytes of buffering. Network Port A connects externally through six dedicated pins. The network port can operate in either half-duplex or full-duplex mode, selected via auto-negotiation. This port requires an accurate 20 MHz clock to generate the 10 Mbits/s serial rate of 10Base-T. The network port contains synchronization circuitry to allow operation from the 20 MHz reference clock while the main system clock runs independently. The network port transmitter proceeds the transmit data automatically with a preamble and start-frame-delimiter, and appends CRC and the end-frame-delimiter after the last byte. Frame transmission starts automatically once the transmit FIFO is full and any interframe gap time or back-off time has expired. Transmission is aborted if a collision is detected, and is retried up to 16 times using the standard random back-off time algorithm. Detection of a collision causes the transmitter to send a 32-bit "jam" pattern of all ones to guarantee that all receivers in the network recognize the collision.

5.2 Features-

The Rabbit 4000 requires no external memory driver or interface-logic. Its 24-bit address bus, 8-bit or 16-bit data bus, three chip-select lines, two output-enable lines, and two write-enable lines can be interfaced directly with up to six memory devices. Up to 1 MB of memory can be accessed directly via the Dynamic C development software, and up to 16 MB can be interfaced with additional software development. A built-in slave port allows the Rabbit 4000 to be used as master or slave in multi-processor systems, permitting separate tasks to be assigned to dedicated processors. An 8-line data port and five control signals simplify the exchange of data between devices. A remote cold boot enables startup and programming via a serial port or the slave port.

The Rabbit 4000 features five 8-bit parallel ports, yielding a total of 40 digital I/O. Six CMOS-compatible serial ports are available. All six are configurable as asynchronous (including output pulses in IrDA format), while four are configurable as clocked serial (SPI) and two are configurable as SDLC/HDLC. The various internal peripherals share the parallel port's I/O pins.

The Rabbit 4000 also offers many specialized peripherals. Two input-capture channels each have a 16-bit counter, clocked by the output of an internal timer, that can be used to capture and measure pulses. These measurements can be extended to a variety of functions such as measuring pulse widths or for baud-rate auto detection. Two quadrature decoder channels each have two inputs, as well as an 8 or 10-bit up/down counter. Each quadrature decoder channel provides a direct interface to optical encoder units. Four independent pulse width modulator (PWM) outputs, each based on a 1024-pulse frame, are driven by the output of a programmable internal timer. The PWM outputs can be filtered to create a 10-bit D/A converter or they can be used directly to drive devices such as motors or solenoids. Two external interrupt vectors can multiplex inputs from up to six external pins. There are numerous timers available for use in the Rabbit 4000. Timer A consists of ten 8-bit counters, each of which has a programmed time constant. Six of them can be cascaded from the primary Timer A counter. Timer B contains a 10-bit counter, two match registers, and two step registers. An interrupt can be generated or the output pin can be updated when the counter reaches a match value, and the match value is then incremented automatically by the step value. Timer C is a 16-bit counter that counts up to a programmable limit. It contains eight match registers, four to set the output of a parallel-port pin and four to reset it. This allows for the creation of PWM signals (both synchronous and variable-phase) and quadrature signals.

The Rabbit 4000 also provides support for protected operating systems. Support for two levels of operation, known as *system* and *user* modes, allow application-critical code to operate in safety while user code is prevented from inadvertently disturbing the setup of the processor. Memory blocks as small as 4KB can be write-protected against accidental writes by user code, and stack over/underflows can be trapped by high-priority interrupts. Security features were also introduced in the Rabbit 4000. Portions of the new instruction set were introduced to dramatically increase encryption algorithm speeds, and 32 bytes of battery-backed onchip-encryption RAM store an encryption key away from prying eyes. The Rabbit 4000 has new peripherals — DMA access and on-chip Ethernet. The Rabbit 4000 supports eight channels of DMA access to external memory, internal I/O addresses, and the auxiliary I/O bus. Directing a DMA channel to or from an internal peripheral such as a serial port or the Ethernet port automatically connects DMA enable signals. Burst size, priority, and guaranteed cycles for the processor are all under program control. The Rabbit 4000 contains a fully featured 10Base-T Ethernet peripheral. Designed to operate with the DMA peripheral, the Ethernet peripheral is fully compliant with the 802.3 Ethernet standards, including support for auto-negotiation, link detection, multicast filtering, and broadcast addresses. All digital components of the 10Base-T MAC and PHY are present inside the Rabbit 4000; all that is needed to interface to an Ethernet network is some simple analog filtering and wave-shaping components.

5.3 Ethernet Cable-

Category 5 cables (**Cat 5**) is a twisted pair cable for carrying signals. This type of cable is used in structured cabling for computer networks such as Ethernet. It is also used to carry other signals such as telephony and video. The cable is commonly connected using punch down blocks and modular connectors. Most Category 5 cables are unshielded, relying on the twisted pair design and differential signaling for noise rejection. Category 5 has been superseded by the **Category 5e** specification.

Category 5 patch cable in T568B wiring

Figure 16 CAT 5 Cable

5.3.1 Cable Standard-

Each of the four pairs in a Cat 5 cable has differing precise number of twists per metre based on prime numbers to minimize crosstalk between the pairs. On average there are 6 twists per 5 centimetres. The pairs are made from 24 gauge (AWG) copper wires within the cables. Although cable assemblies containing 4 pairs are common, Category 5 is not limited to 4 pairs. Backbone applications involve using up to 100 pairs.This use of balanced lines helps preserve a high signal-to-noise ratio despite interference from both external sources and crosstalk from other pairs. Category 5 cabling is most commonly used for faster Ethernet networks, such as 100BASE-TX and 1000BASE-T.

The cable is available in both stranded and solid conductor forms. The stranded form is more flexible and withstands more bending without breaking and is suited for reliable connections with insulation piercing connectors, but makes unreliable connections in insulation-displacement connectors (IDCs The solid form is less expensive and makes reliable connections into insulation displacement connectors, but makes unreliable connections in insulation piercing connectors.

5.3.2 Performance of CAT5-

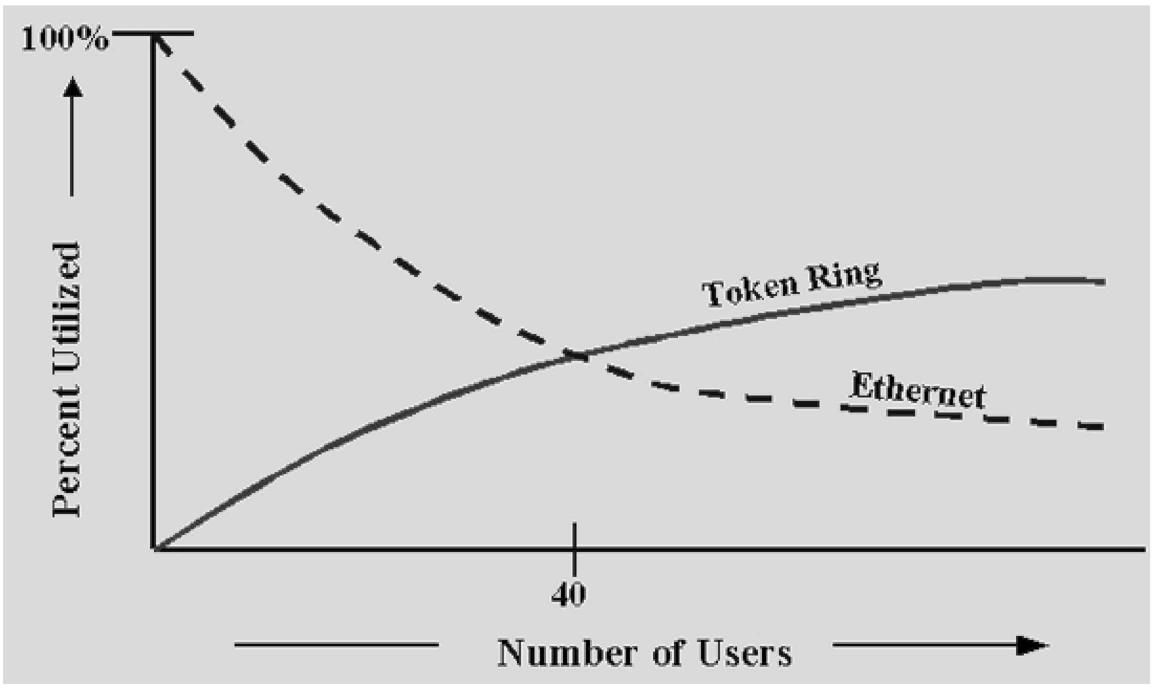


Figure 17 Performance of CAT 5

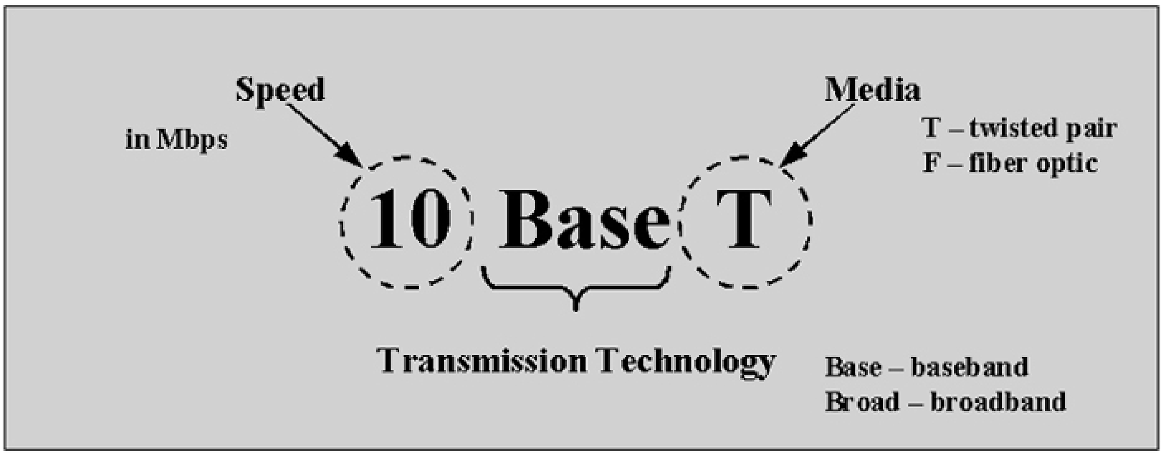
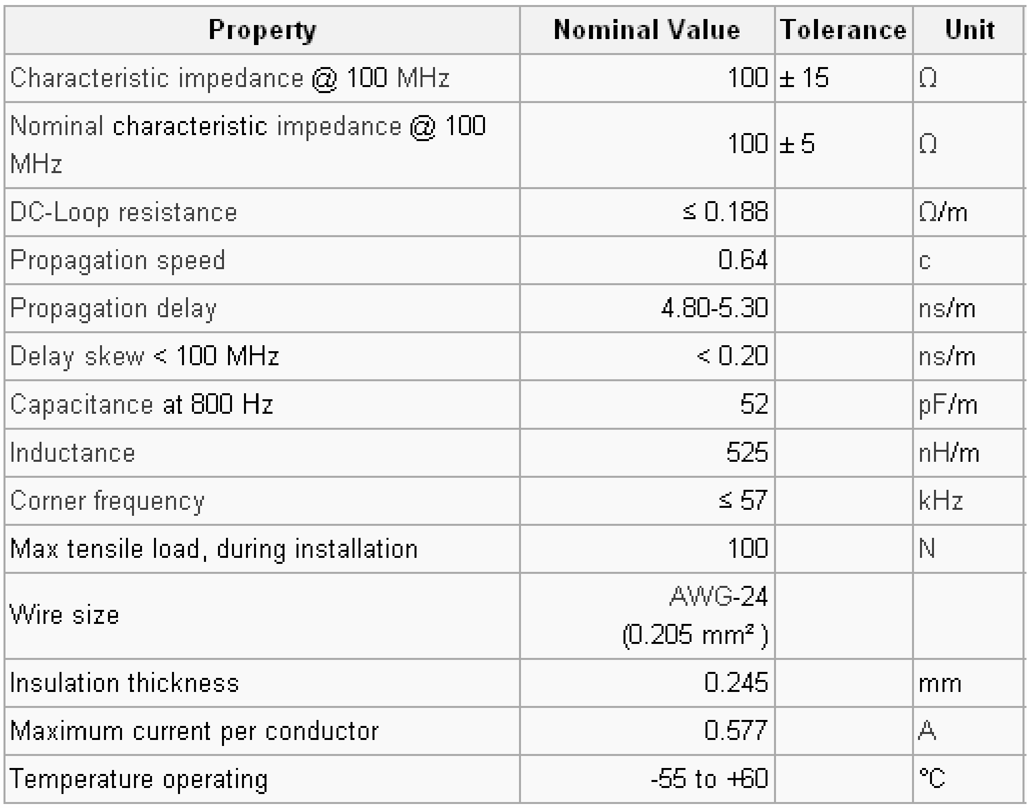


Figure 18 Meaning of 10BaseT [Transmission Technology]

5.3.3 Specifications of CAT5-



Property	Nominal Value	Tolerance	Unit
Characteristic impedance @ 100 MHz	100	± 15	Ω
Nominal characteristic impedance @ 100 MHz	100	± 5	Ω
DC-Loop resistance	≤ 0.188		Ω/m
Propagation speed	0.64		c
Propagation delay	4.80-5.30		ns/m
Delay skew < 100 MHz	< 0.20		ns/m
Capacitance at 800 Hz	52		pF/m
Inductance	525		nH/m
Corner frequency	≤ 57		kHz
Max tensile load, during installation	100		N
Wire size	AWG-24 (0.205 mm^2)		
Insulation thickness	0.245		mm
Maximum current per conductor	0.577		A
Temperature operating	-55 to +60		°C

Figure 19 Specifications of CAT 5

5.4 Personal Computer with processor Pentium 4-

Pentium 4 and new Celeron processors use Intel’s seventh generation architecture, also called Netburst. Its overall look you can see in Figure. Don’t get scared. We will explain deeply what this diagram is about.

In order to continue, however, you need to have read our tutorial “How a CPU Works”. In this tutorial we explain the basics about how a CPU works. In the present tutorial we are assuming that you have already read it, so if you didn’t, please take a moment to read it before continuing, otherwise you may find yourself a little bit lost. Actually we can consider the present tutorial as a sequel to our How a CPU Works tutorial. Here are the basic differences between Pentium 4 architecture and the architecture from other CPUs:

- Externally, Pentium 4 transfers four data per clock cycle. This technique is called QDR (Quad Data Rate) and makes the local bus to have a performance four times its actual clock rate, see table below. In Figure this is shown on “3.2 GB/s System Interface”; since this slide was produced when the very first Pentium 4 was released, it mentions the “400 MHz” system bus.

- The data path between the L2 memory cache ("L2 cache and control" in Figure 1) and L1 data cache ("L1 D-Cache and D-TLB" in Figure 1) is 256-bit wide. On previous processors from Intel this data path was of only 64 bits. So this communication can be four times faster than processors from previous generations when running at the same clock. The data path between L2 memory cache ("L2 cache and control" in Figure 1) and the pre-fetch unit ("BTB & I-TLB" in Figure 1), however, continues to be 64-bit wide.
- The L1 instruction cache was relocated. Instead of being before the fetch unit, the L1 instruction cache is now after the decode unit, with a new name, "Trace Cache". This trace cache can hold up to 12 K microinstructions. Since each microinstruction is 100-bit wide, the trace cache is of 150 KB (12 K x 100 / 8). One of the most common mistakes people make when commenting Pentium 4 architecture is saying that Pentium 4 doesn't have any instruction cache at all. That's absolutely not true. It is there, but with a different name and a different location.
- On Pentium 4 there are 128 internal registers, on Intel's 6th generation processors (like Pentium II and Pentium III) there were only 40 internal registers. These registers are in the Register Renaming Unit (a.k.a. RAT, Register Alias Table, shown as "Rename/Alloc" in Figure 1).
- Pentium 4 has five execution units working in parallel and two units for loading and storing data on RAM memory.

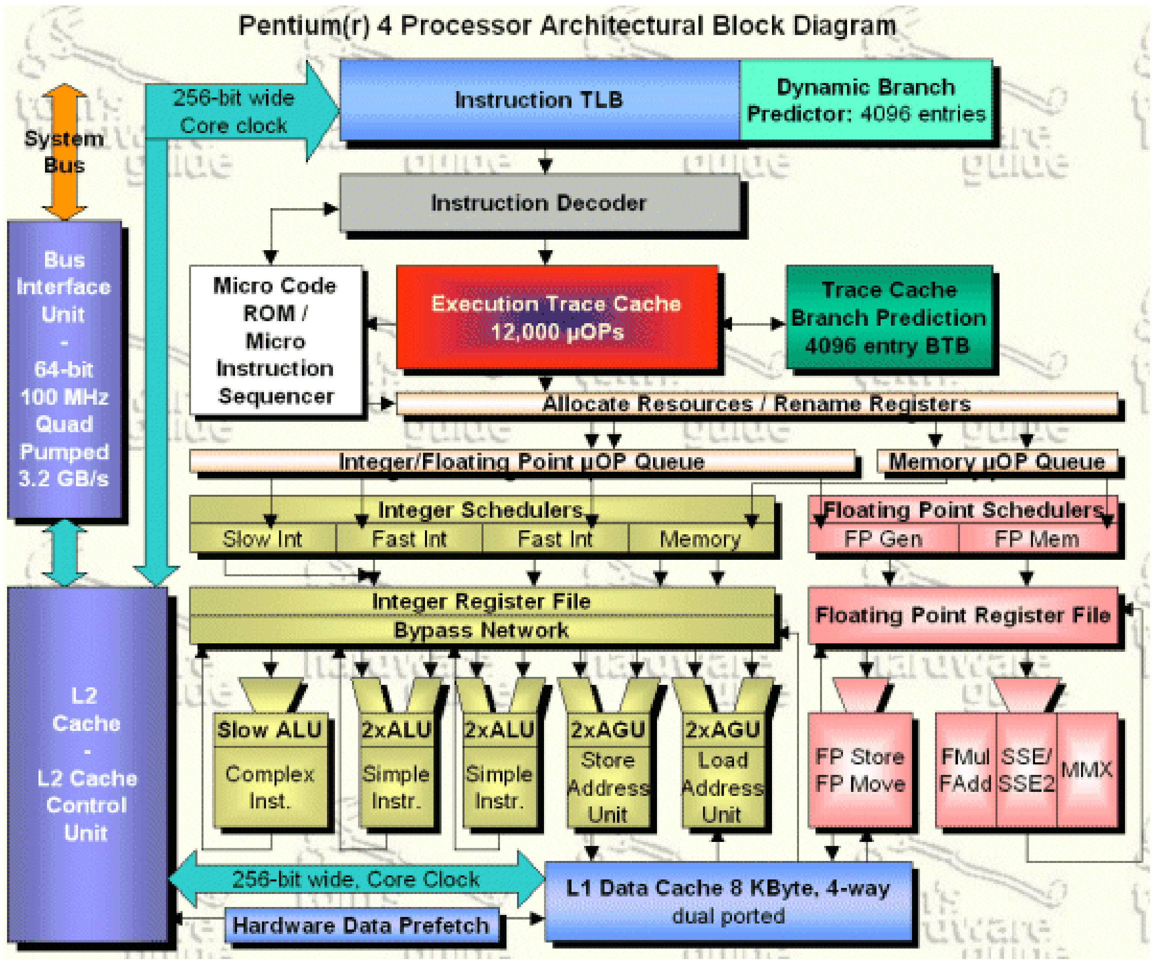

Pentium(r) 4 Processor Architectural Block Diagram
System Bus
256-bit wide Core clock
Instruction TLB
Dynamic Branch Predictor: 4096 entries
Instruction Decoder
Bus Interface Unit - 64-bit 100 MHz Quad Pumped 3.2 GB/s
Micro Code ROM / Micro Instruction Sequencer
Execution Trace Cache 12,000 µOPs
Trace Cache Branch Prediction 4096 entry BTB
Allocate Resources / Rename Registers
Integer/Floating Point µOP Queue
Memory µOP Queue
Integer Schedulers
Slow Int
Fast Int
Fast Int
Memory
Floating Point Schedulers
FP Gen
FP Mem
Integer Register File
Bypass Network
Floating Point Register File
L2 Cache - L2 Cache Control Unit
Slow ALU Complex Inst.
2xALU Simple Instr.
2xALU Simple Instr.
2xAGU Store Address Unit
2xAGU Load Address Unit
FP Store FP Move
FMul FAdd
SSE/ SSE2
MMX
256-bit wide, Core Clock
Hardware Data Prefetch
L1 Data Cache 8 KByte, 4-way dual ported

Figure 20 Pentium 4 Block Diagram

Real Clock	Performance	Transfer Rate
100 MHz	400 MHz	3.2 GB/s
133 MHz	533 MHz	4.2 GB/s
200 MHz	800 MHz	6.4 GB/s
266 MHz	1,066 MHz	8.5 GB/s

Table 2 QDR of P4

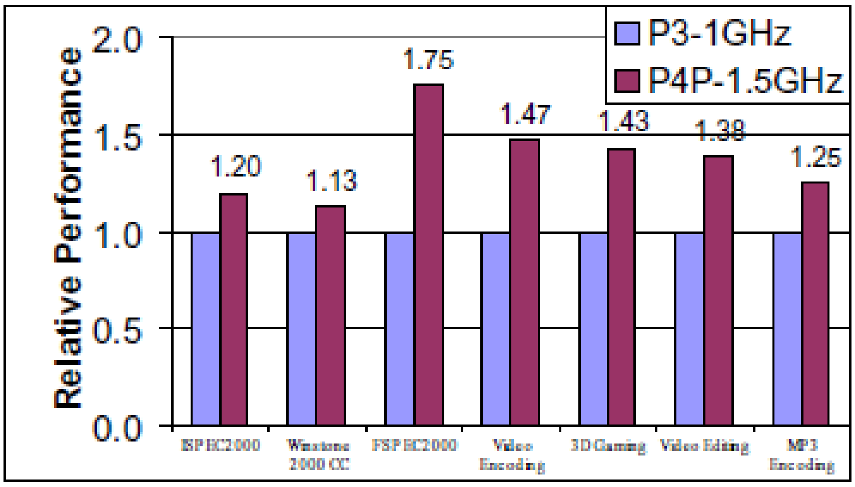


Figure 21 Performance comparison of P4 with P3

<u>5.4.1 Advanced L2 Cache-</u>

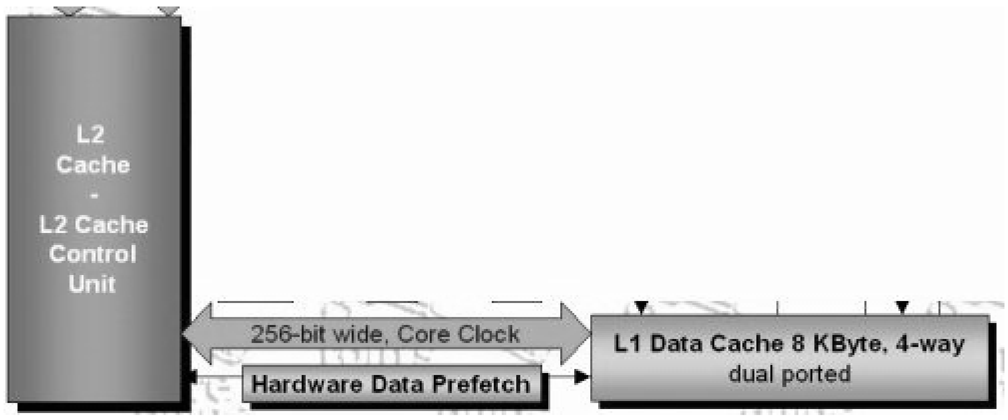

L2 Cache - L2 Cache Control Unit
256-bit wide, Core Clock
Hardware Data Prefetch
L1 Data Cache 8 KByte, 4-way dual ported

Figure 22 Advanced L2 Cache of P4

The L2-cache is called also ‘Advanced Transfer Cache’. With 256 KB its size is identical to the L2-cache of Pentium III and both are 8-way associative as well. Pentium 4’s L2-cache is using 128-byte cache lines, which are divided in two 64-byte pieces. While Pentium III is equipped with a 16KB L1 cache for instructions and a 16KB L1 cache for data, there is only an 8 KB small data L1 cache in Pentium 4, while a feature called ‘Execution Trace Cache’ replaces the L1 instruction cache of Pentium III. Intel was probably forced to reduce the size of the L1 data cache down to only 8 KB, to enable its extremely low latency of only 2 clock cycles. It results in an overall improved read latency.The L1 data cache of Pentium 4 is 4-way set associative and uses 64-byte cache-lines. The dual-port architecture allows one load and one store operation per clock.

5.4.2 Branch Prediction & Trace Cache-

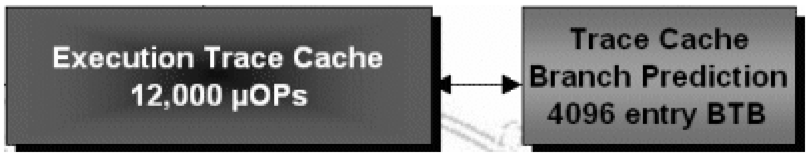


Figure 23 Branch Prediction & Trace Cache of P4

5.4.3 Bus Interface-

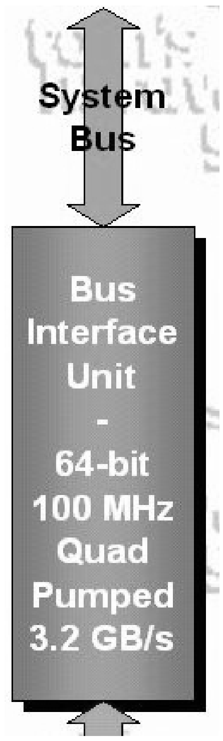


Figure 24 Bus Interface of P4

The first new feature seen by code or data as it enters Pentium 4 is the new system bus. Pentium 4's system bus is only clocked at 100 MHz and also 64-bit wide, but it is 'quad-pumped', using the same principle as AGP4x. Thus it can transfer 8 byte * 100 milions/s * 4 = 3,200 MB/s.

The new bus of Pentium 4 enables it to exchange data with the rest of the system faster than any other x86-processor, thus removing one important bottleneck that Pentium 3 was suffering from. The Intel's new 850 chipset for Pentium 4, is using two Rambus channels and therefore the expensive and unpopular RDRAM. However, these two Rambus channels are able to deliver the same data bandwidth as Pentium 4's new bus (3,200 MB/s), making them a perfect match at least on paper.

5.4.4 Instruction Decoder & Trace Cache-

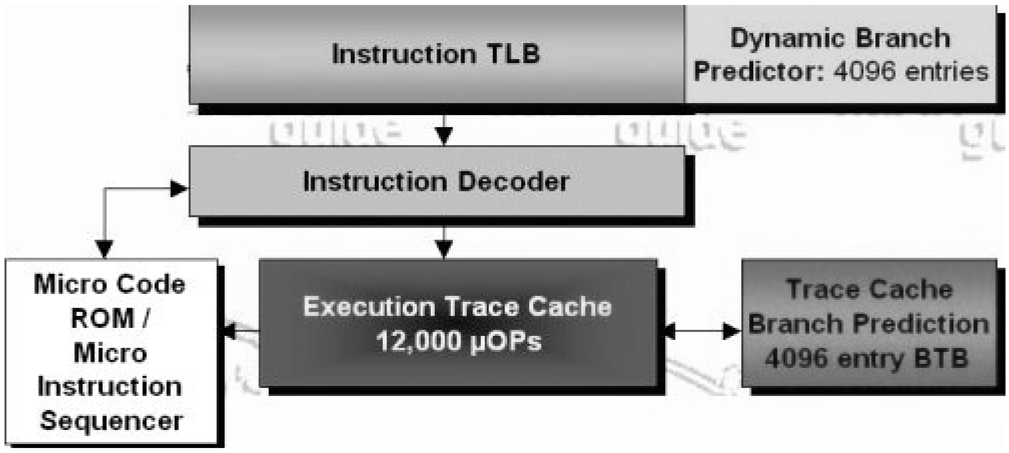

Instruction TLB
Dynamic Branch Predictor: 4096 entries
Instruction Decoder
Micro Code ROM / Micro Instruction Sequencer
Execution Trace Cache 12,000 µOPs
Trace Cache Branch Prediction 4096 entry BTB

Figure 25 Instruction Decoder & Trace Cache

5.4.5 Pipeline-

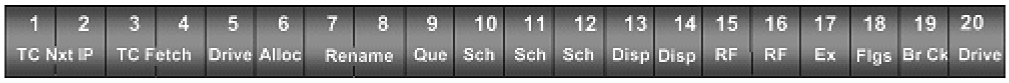


Figure 26 Pipelining in P4

One of the most well known features of the new Pentium 4 is its extremely long pipeline. While the pipeline of Pentium III has 10 stages and the one of Athlon 11, Pentium 4 has no less than 20 stages. The reason for the longer pipeline is Intel’s wish of Pentium 4 to deliver highest clock rates. The smaller or shorter each pipeline stage, the fewer transistors or ‘gates’ it needs and the faster it is able to run. However, there is also one big disadvantage to long pipelines. As soon as it turns out at the end of the pipeline that the software will branch to an address that was not predicted, the whole pipeline needs to be flushed. The longer the pipeline the more ‘in-flight’ instructions will be lost and the longer it takes until the pipeline is filled again.

Intel is proud to announce that the Pentium 4 pipeline can keep up to 126 instructions ‘in-flight’, among them up to 48 load and 24 store operations. The improved trace cache branch prediction unit described above is supposed to ensure that flushes of this long pipeline are only rare occasions.

The stuff that happens in the trace cache, as mentioned above, only represents the first five stages of the pipeline of the Pentium 4. What follows is:

- ✓ allocate resources
- ✓ register renaming
- ✓ write into the microOp queue

- ✓ write into the schedulers and compute dependences
- ✓ dispatch microOP's to their execution units
- ✓ read register file (to ensure that the correct ones of the 128 all-purpose register files are used as the register(s) for the actual instruction).

After that comes the actual execution of the microOP.

5.4.6 Rapid Execution Engine-

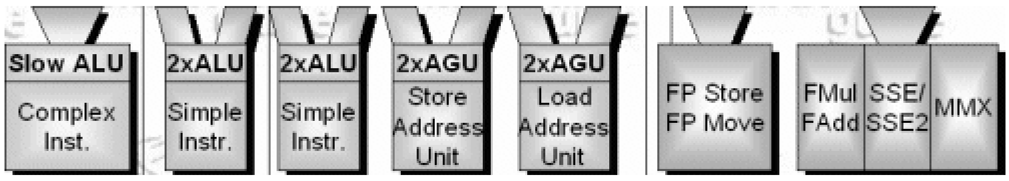


Figure 27 Rapid Execution Engine

The above picture is actually showing all execution units of Pentium 4, including the "Rapid Execution Engine' as well as the 'not-so-rapid' execution units. While Intel is only talking about the four fast execution units, the other four are the actual units that are responsible for Pentium 4's peculiar behavior in the benchmarks.

Basic part of the 'Rapid Execution Engine' is the two 'double-pumped' ALUs and AGUs. Each of the four is said to be clocked with double the processors clock, because they can receive microOPs every half clock. Simple microOPs that can be processed by the Rapid Execution Engine are executed in a half clock, which is obviously a very good thing.

The story looks a lot different for the instructions that cannot be processed by the rapid execution units. Those instructions or microOps need to use the one and only 'Slow ALU', which is not 'doubling pumped'.

The majority of instructions need to use this path, which obviously sounds scary.SSE2 – The New Double Precision Streaming SIMD Extensions

- ✓ 144 new instructions
- ✓ 4 single precision FP values (SSE)
- ✓ 2 double precision FP values (SSE2)
- ✓ 16 byte values (SSE2)
- ✓ 8 word values (SSE2)
- ✓ 4 double word values (SSE2)
- ✓ 2 quad word values (SSE2)
- ✓ 1 128-bit integer value (SSE2)

5.5 LCD Display-

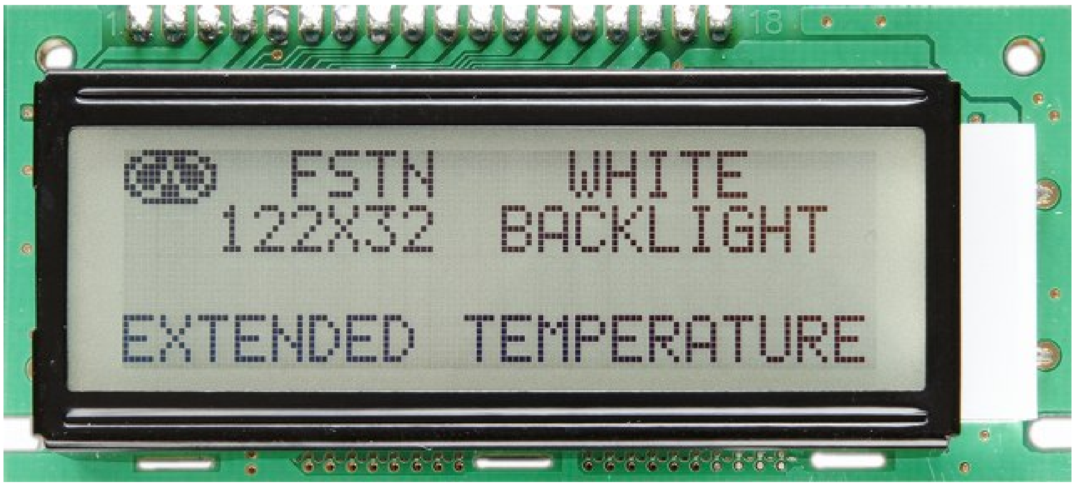


Figure 28 LCD 122x32 Display

5.5.1 Features-

Type: Graphic

Display format: 122 x 32 dots

Built-in controller: Epson SED1520 (or equivalent)

Duty cycle: 1/32

Available for internal oscillation 2 kHz

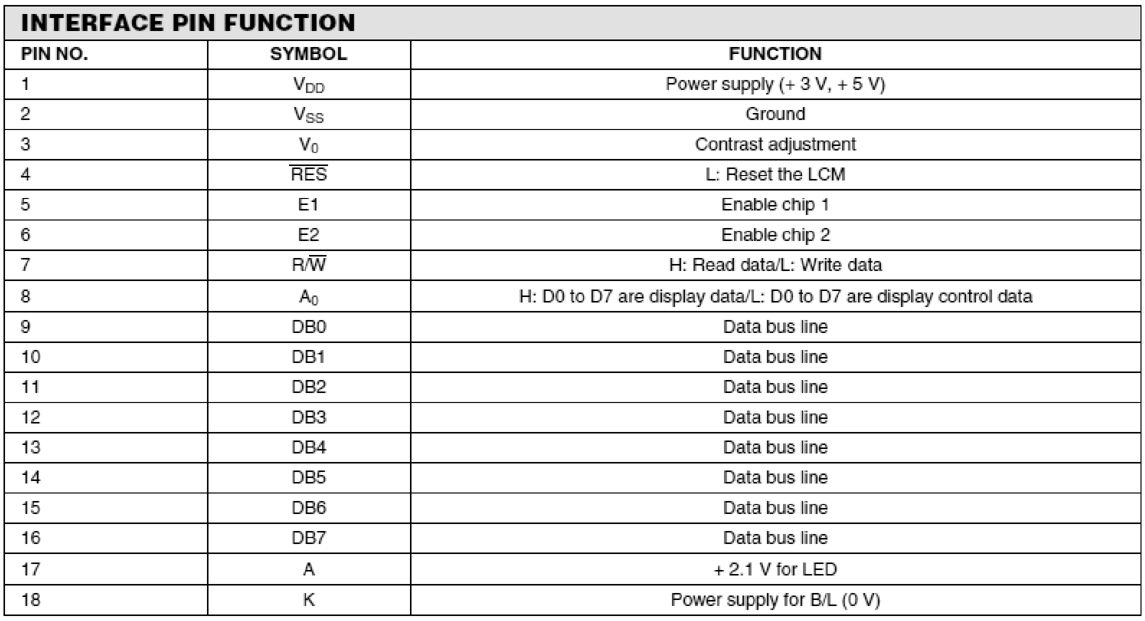
+ 5 V power supply only

The feature of LCD-122H032G is same as LCD-122H032B

Chinese version: LCD-122H032M

Compliant to RoHS directive 2002/95/EC

5.5.2 LCD Pin Description-

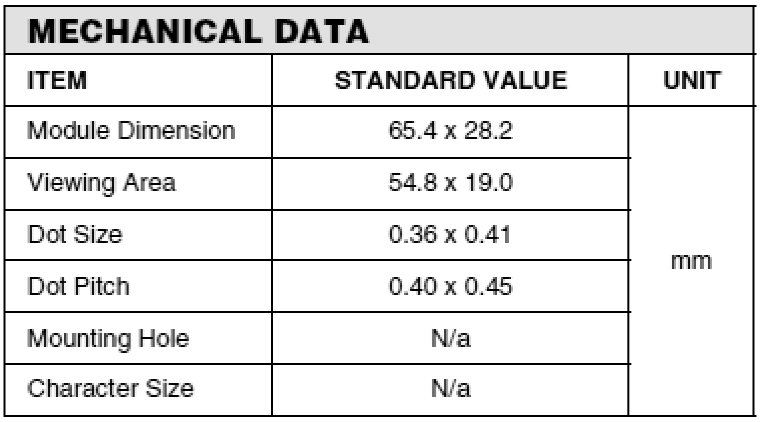


INTERFACE PIN FUNCTION

PIN NO.	SYMBOL	FUNCTION
1	V_{DD}	Power supply (+ 3 V, + 5 V)
2	V_{SS}	Ground
3	V_0	Contrast adjustment
4	$\overline{RES}$	L: Reset the LCM
5	E1	Enable chip 1
6	E2	Enable chip 2
7	$R/\overline{W}$	H: Read data/L: Write data
8	A_0	H: D0 to D7 are display data/L: D0 to D7 are display control data
9	DB0	Data bus line
10	DB1	Data bus line
11	DB2	Data bus line
12	DB3	Data bus line
13	DB4	Data bus line
14	DB5	Data bus line
15	DB6	Data bus line
16	DB7	Data bus line
17	A	+ 2.1 V for LED
18	K	Power supply for B/L (0 V)

Table 3 LCD 122x 32 Pin Descriptions

5.5.3 Mechanical [Physical] Dimensions-



MECHANICAL DATA

ITEM	STANDARD VALUE	UNIT
Module Dimension	65.4 x 28.2	mm
Viewing Area	54.8 x 19.0	
Dot Size	0.36 x 0.41	
Dot Pitch	0.40 x 0.45	
Mounting Hole	N/a	
Character Size	N/a	

Table 4 Physical Dimensions of LCD

5.5.4 Electrical Characteristics of LCD-

ELECTRICAL CHARACTERISTICS

ITEM	SYMBOL	CONDITION	STANDARD VALUE			UNIT
			MIN.	TYP.	MAX.	
Input Voltage	V_{DD}	$V_{DD} = +5\ V \pm 1\ V$	4.5	5.0	5.5	V
Supply Current	I_{DD}	$V_{DD} = +5\ V$	-	1.0	1.4	mA
Recommended LC Driving Voltage for Normal Temperature Version Module	V_{DD} to V_0	- 20 °C	4.7	4.9	5.1	V
		0 °C	4.5	4.7	4.9	
		25 °C	4.3	4.5	4.7	
		50 °C	4.2	4.3	4.5	
		70 °C	4.0	4.1	4.3	
LED Forward Voltage	V_F	25 °C	1.7	2.1	2.5	V
LED Forward Current	I_F	25 °C	-	100	200	mA
EL Power Supply Current	I_{EL}	$V_{EL} = 110\ V_{AC}$, 400 Hz	-	-	5.0	mA

Table 5 Electrical Characteristics of LCD

5.5.5 Internal Block Diagram of LCD-

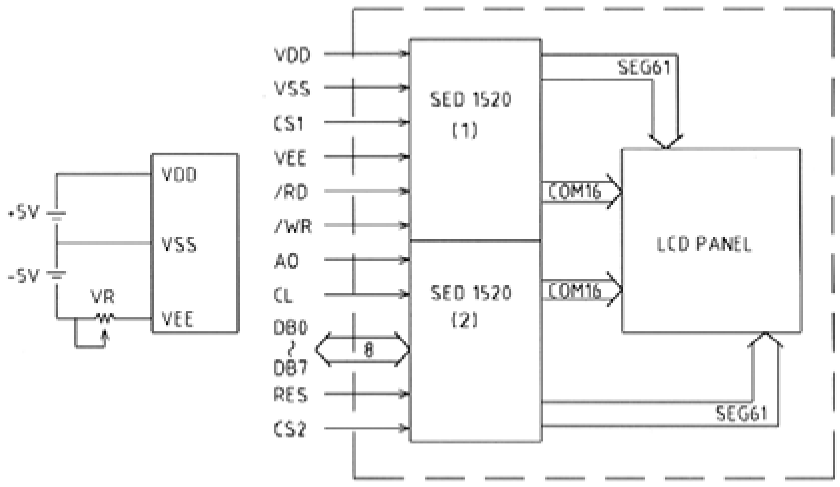


Figure 29 Internal Block Diagram of LCD

5.5.6 Interfacing of LCD Parallel Ports or Details about parallel Port A-

As discussed earlier that in Rabbit 4000 Microprocessor there are five Parallel Ports named as **Parallel Port A**, **Parallel Port B**, **Parallel Port C**, **Parallel Port D** and **Parallel Port E**. The below block diagram will describe how various parallel ports are I/O Ports are connected with Rabbit 4000 Microprocessor in RCM 4000 prototype board. Each parallel Port has 8 pins. From Pin 0 to Pin 7. Parallel ports have some dual characteristics. But these ports are used as an I/O ports i.e. we can attach various external Hardwares in RCM4000 Prototype board for our Project purpose. To configure those Hardwares and to make the proper use of those Hardwares we have to program those External equipments through various parallel Ports by Dynamic C. There are various registers in those Parallel Ports which are helping us to configure those Parallel Ports and send or get some data to or from those External Hardwares.

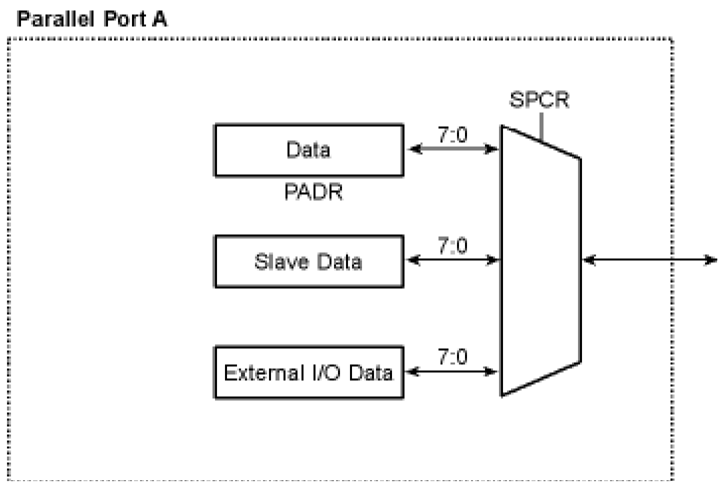


Figure 30 Parallel Port A in Rabbit Processor

CHAPTER 6 - SOFTWARE DESCRIPTION

6.1 Dynamic C-

Dynamic C is an integrated development system for writing embedded software. It is designed for use with Rabbit controllers and other controllers based on the Rabbit microprocessor.

6.1.1 The Nature of Dynamic C-

Dynamic C integrates the following development functions into one program:

- Editing
- Compiling
- Linking
- Loading
- . Debugging

Dynamic C has an easy-to-use, built-in, full-featured text editor. Dynamic C programs can be executed and debugged interactively at the source-code or machine-code level. Pull-down menus and keyboard shortcuts for most commands make Dynamic C easy to use.

Dynamic C also supports assembly language programming. It is not necessary to leave C or the development system to write assembly language code. C and assembly language may be mixed together.

Debugging under Dynamic C includes the ability to use printf commands, watch expressions, breakpoints and stack tracing. Watch expressions can be used to compute C expressions involving the target's program variables or functions. Watch expressions can be evaluated while stopped at a breakpoint or while the target is running its program. Stack tracing shows function call sequences and parameter values.

Dynamic C provides extensions to the C language (such as shared and protected variables, costatements and cofunctions) that support real-world embedded system development. Dynamic C supports cooperative and preemptive multitasking.

Dynamic C comes with many function libraries, all in source code. These libraries support real-time programming, machine level I/O, and provide standard string and math functions.

6.1.2 Speed-

Dynamic C compiles directly to memory. Functions and libraries are compiled and linked and downloaded on-the-fly. On a fast PC, Dynamic C might load 30,000 bytes of code in five seconds at a baud rate of 115,200 bps.

5.1.3 New Features from ANSI C-

The following feature was introduced with Dynamic C 10.62:

• Function pointer parameter list checking: Function pointers may now contain a parameter list, and the compiler will check the parameters and perform automatic type promotion when a function is called through the function pointer.

The following features were introduced with Dynamic C 10.64:

• "File Scoping: In projects using multiple .C files, each C source file now defines a new scope separate from all other C files. Using the static keyword ensures that file-scope variables and functions are visible only within the file in which they are defined. Omitting the static or using extern will link those symbols to other files where they are used. The new functionality obeys the ANSI-C rules for file scoping for .C and .H files. The Dynamic C extension #use (which works with .LIB files) works with the new scoping with some important caveats noted in section 4.7 "Libraries".

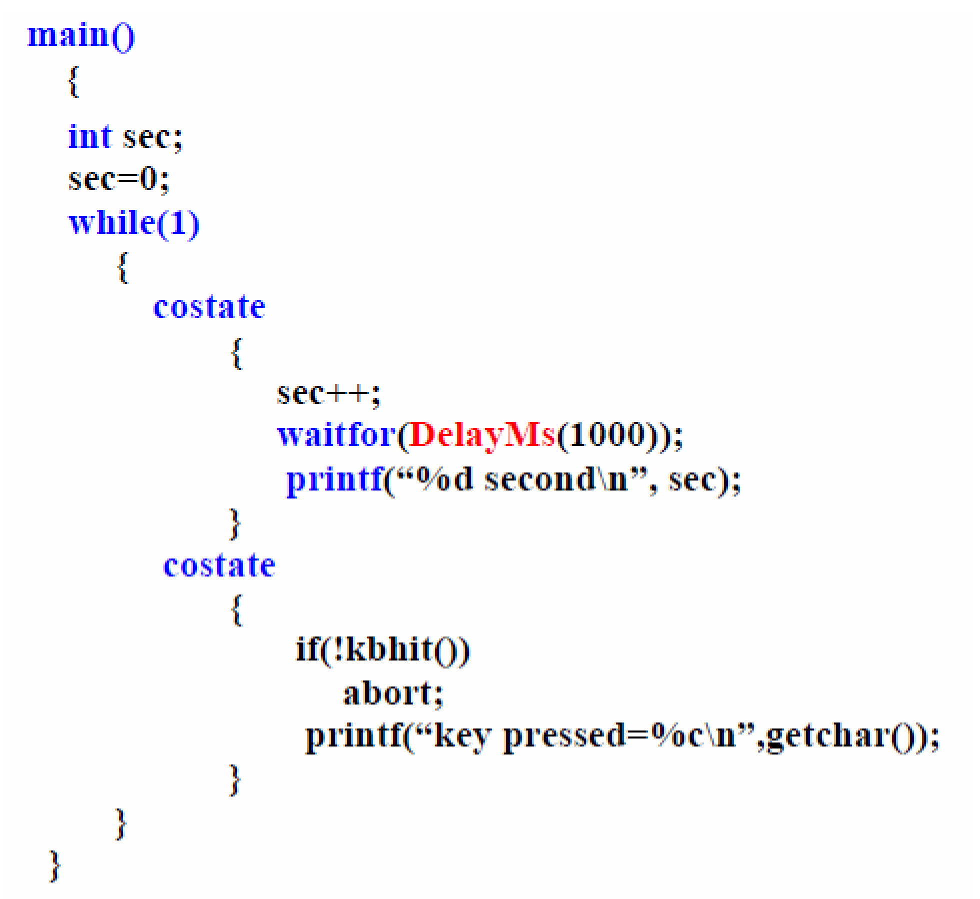
• "Nested/block Scoping: Nested, or block, scoping allows the declaration of variables within a curly brace delimited block. Previously, Dynamic C only allowed declarations at the beginning of a function block (as defined by the original K&R C specification). In addition, #asm assembly blocks now each have their own local (function) block scope preventing name collisions between global and local assembly labels

• "Const Correctness: Dynamic C now handles the functionality of the const keyword as it is defined in the ANSI-C89/ISO-C90 specification. The differences between the older Dynamic C functionality and the new ANSI functionality.

• "signed Keyword: The signed keyword is now supported.   Variables of type char can also now be signed; previously, char variables could only be unsigned.

6.1.4 Enhancements of Dynamic C-

Many other important features have been added to Dynamic C for making it campatible with the Embedded System. Some of these Enhancements are:
1. Function Changing, a concept very unique to Dynamic C, allows special segments of code to be embedded within one or more functions. When a named function chain executes, all the segments belonging to that chain execute. Function chains allow software to perform initialization, data recovery, or other kinds of tasks on request. 2. Cooperative Multitasking is a brand new idea to Dynamic C. It allows c Cooperative Multitasking is a way to perform different tasks at virtually the same time. Unlike Preemptive Multitasking, in cooperative multitasking variables can be shared between different tasks without taking elaborate precautions. Cooperative multitasking also takes advantage of the natural delays that occur in most tasks to more efficiently use the available processor time.
Dynamic C has language extensions to support cooperative multitasking. To implement it they have COSTATEMENTs and COFUNCTIONs.
Costatements allow cooperative, parallel process to be simulated in a single program, while Costatements allow only cooperative process to be simulated in a single program.
An example of cooperative Multitasking by the costatement function is shown below:



```
main()
   {
   int sec;
   sec=0;
   while(1)
       {
         costate
               {
                  sec++;
                  waitfor(DelayMs(1000));
                   printf("%d second\n", sec);
               }
          costate
               {
                    if(!kbhit())
                        abort;
                     printf("key pressed=%c\n",getchar());
               }
       }
 }
```

Figure 31 General Dynamic C example

Here in the above example the first cosatement will continue to print the the value of the variable sec by incrementing the value by 1 at every 1 second interval. When the Program Counter got the statement Delay then it will nump from that particular costatement and will run the 2nd costatement. In the 2nd costatement it will check whether any key is being pressed in the 1 second. If not then it will abort from that costatement and will jump to the immediate next line of delay function of 1st costatement and will print the value of sec. But if any key is being pressed in the time of checking the 2nd costatement condition then it will also print that particular character. So by using the costatement and Delay function we can run two tasks in a virtually same time. That is the one of the main advantage of Cooperative Multitasking.

3. Slice Statements allow preemptive process in a single program.

4. Dynamic C supports **Embedded Assembly Code** and **stand-alone Assembly Code.**

5. Dynamic C has keywords that help protect data shared between different context or stored in battery-backed memory.

6. Dynamic C has a set of features that allow the Programmer to make the fullest use of **xmem**. (Extended Memory). Before launching the Dynamic C 10 version 10.21 the compiler supported a 1 MB physical address space. But after launching the version 10.21 the compiler now supports up to the 16 MB of Physical memory; up to 16 MB can be used for data and up to the 1 MB can be used for codes.
Normally Dynamic C takes care of memory management, but there are instances where the programmer will want to take control of it. Dynamic C has various keywords and directives which are helping us to put code and data in proper place. Those keywords will be discussed later.
Here in time of writing my Project Report i have assumed that we all know Standard C language or Traditional C language very well. So here I am not discussing about those features which is totally similar to C like **variable declaration and variable names; else-if statement, various type of variables(int, float, char) looping, Functions, Structures, pointers** etc. I will in brief discuss about those features and keywords which is not present in Traditional C language i.e. which are brand new concept in Dynamic C.

6.1.5 Different keywords in Dynamic C-

A keyword is a reserved word in any language like C that represents a basic C construct. It can not be used for any other purposes. If we are using those reserved words or key words for other purposes then it will show some errors. In Dynamic C one can find that it is enriched by various key words for various functions. But here we will not discuss about the all. We will only discuss in brief about those key words which we have used in writing our Program for our project. **1. abort** is used to jump out of a costatement. in time of running a program if the compiler gets any abort keyword in a costatement then it will forcefully jumps out of that costatement and will go to the other costatements or simple statements. **2. auto** is used before declaring a variable. A function's local variable is located on the system stack and exists as long as the function call does.

3. break is used to jump out of a loop, if, or case statement.

4. case identifies the next case in a switch statement.

5. char used to declare a variable or array element as an unsigned 8-bit character.

6. const is used to declare that a value will be stored in Flash, thus making it unavailable for modification. It is a type qualifier and may be used with any static or global type specifier. (char, int, float, etc)

7. continue is used to skip to the next iteration of a loop.

8. costate is indicating the beginning of a costtaement. Name can be preseor absent.

9. do indicates the beginning of a do-while loop. The statement must have semicolon at the end.

10. else indicates the false or Zero branch of an if statement.

11. extern indicates that a variable is defined in the BIOS, later in a library file, or in another library file. Its main use is in module headers.

12. float declares variables, function return values, or arrays, as 32-bit floating point.

13. for indicates the beginning of a for loop. A for loop has an initializing expression, a expression and an increment or decrement expression.

14. if indicates the beginning of an if statement.

15. int declares variables, function return values, or array elements to be 16-bit integers. If nothing else specified, int implies a 16-bit signed integer.

16. long declares variables, function return values, or array elements to be 32-bit integers. If nothing else specified, long implies a signed integer.

17. main indicates the main() function. All programs start at the beginning of main() function. It is actually not a key word. It is a function name.

18. sizeof is a built-in function that returns the size in bytes of a variable, array, structure, union, or a data type.

19. static declares a local variable to have a permanent fixed location in memory.

20. typedef is proving a way to create new names for existing data types.

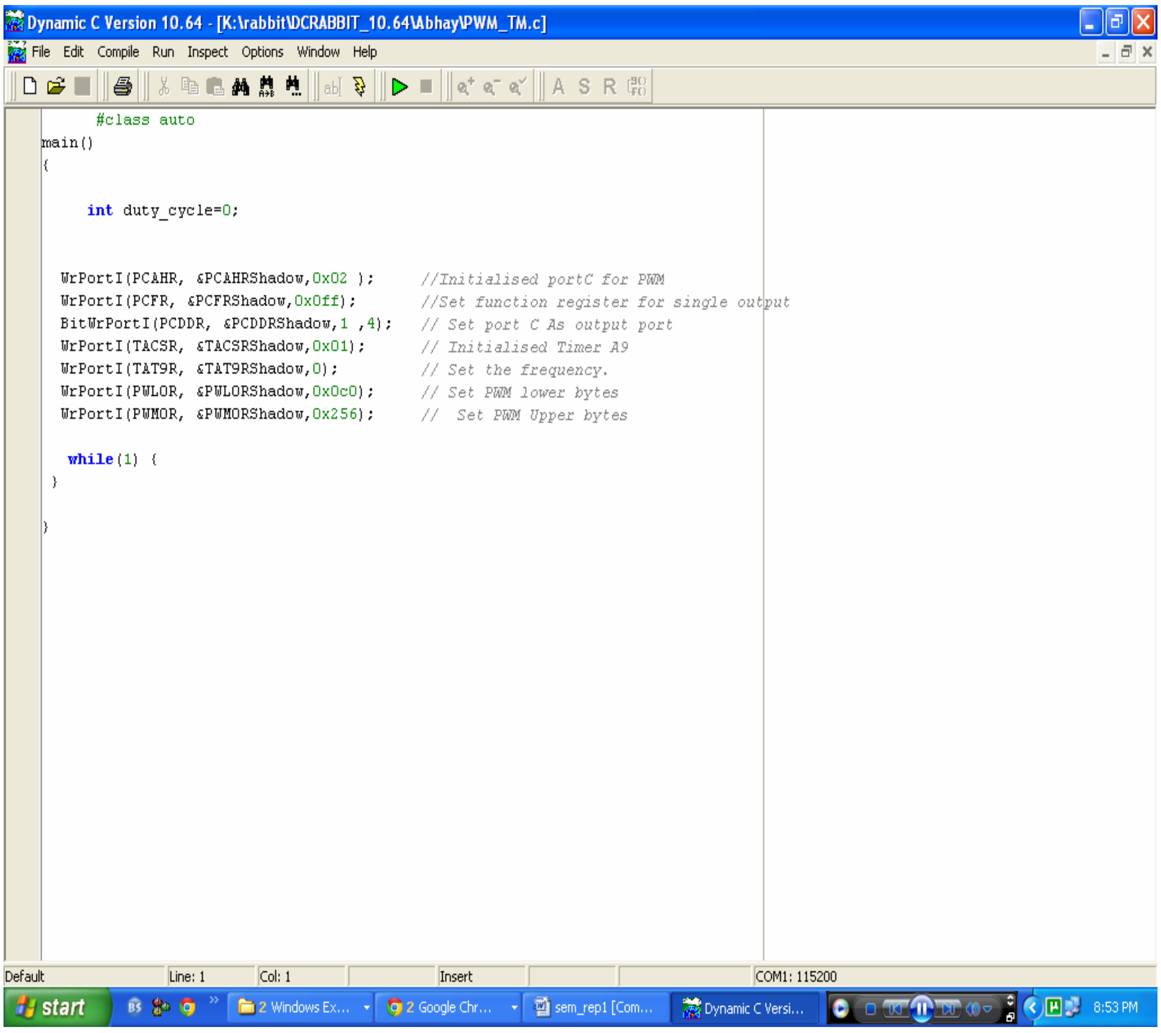
21. unsigned declares a variable or array to be unsigned. if nothing is specified in declaration, unsigned means 16-bit unsigned integer.

22. waitfor used in a costatement or cofunction. This keyword identifies a point of suspension pending the outcome of a condition, completion or an event, or some other delay.

23. while identifies the beginning of a while loop. A while loop tests at the beginning and may execute zero or more times.

24. xmem indicates that a function is to be placed in extended memory. This keyword is semantically meaningful in function prototypes.

25. <u>void</u> keyward conforms to Ansi C. Thus, it can be used in three ways: parameter List, Pointer to void and return type.


Dynamic C Version 10.64 - [K:\rabbit\DCRABBIT_10.64\Abhay\PWM_TM.c]
File Edit Compile Run Inspect Options Window Help
A S R

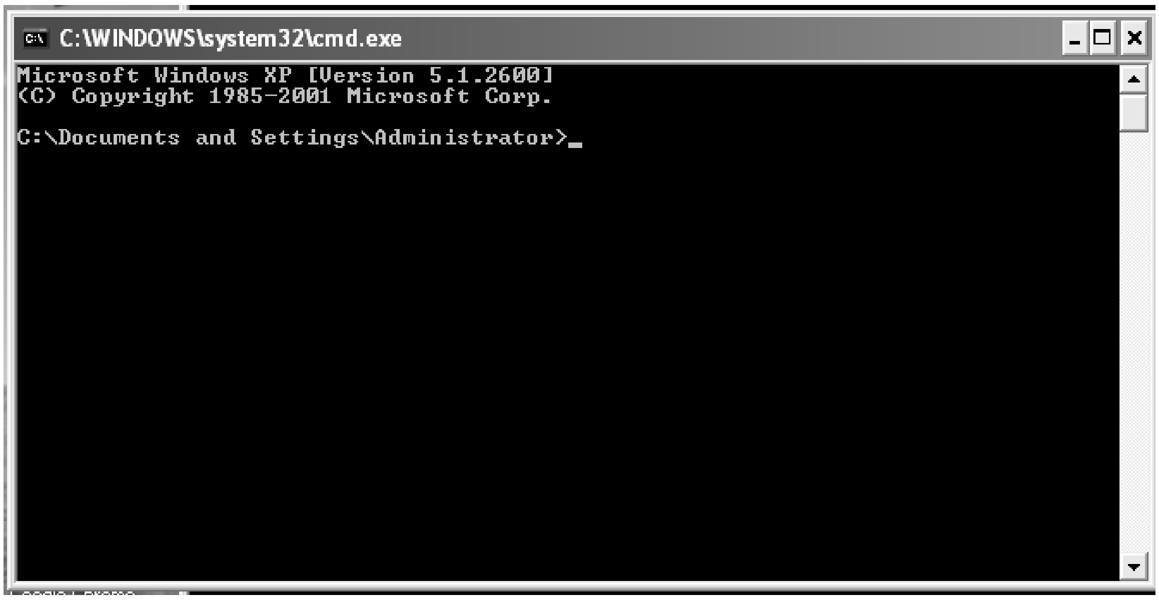
```
      #class auto
main()
{

    int duty_cycle=0;


  WrPortI(PCAHR, &PCAHRShadow,0x02 );      //Initialised portC for PWM
  WrPortI(PCFR, &PCFRShadow,0x0ff);        //Set function register for single output
  BitWrPortI(PCDDR, &PCDDRShadow,1 ,4);    // Set port C As output port
  WrPortI(TACSR, &TACSRShadow,0x01);       // Initialised Timer A9
  WrPortI(TAT9R, &TAT9RShadow,0);          // Set the frequency.
  WrPortI(PWLOR, &PWLORShadow,0x0c0);      // Set PWM lower bytes
  WrPortI(PWMOR, &PWMORShadow,0x256);      //  Set PWM Upper bytes

   while(1) {
 }

}
```

Figure 32 Dynamic C Look

6.2 Command Prompt-

It is present in Start->All programs->Accessories->Command Prompt

It is used when we program the Rabbit Processor, we can check the output on this.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>_
```

Figure 33 Command Prompt Look

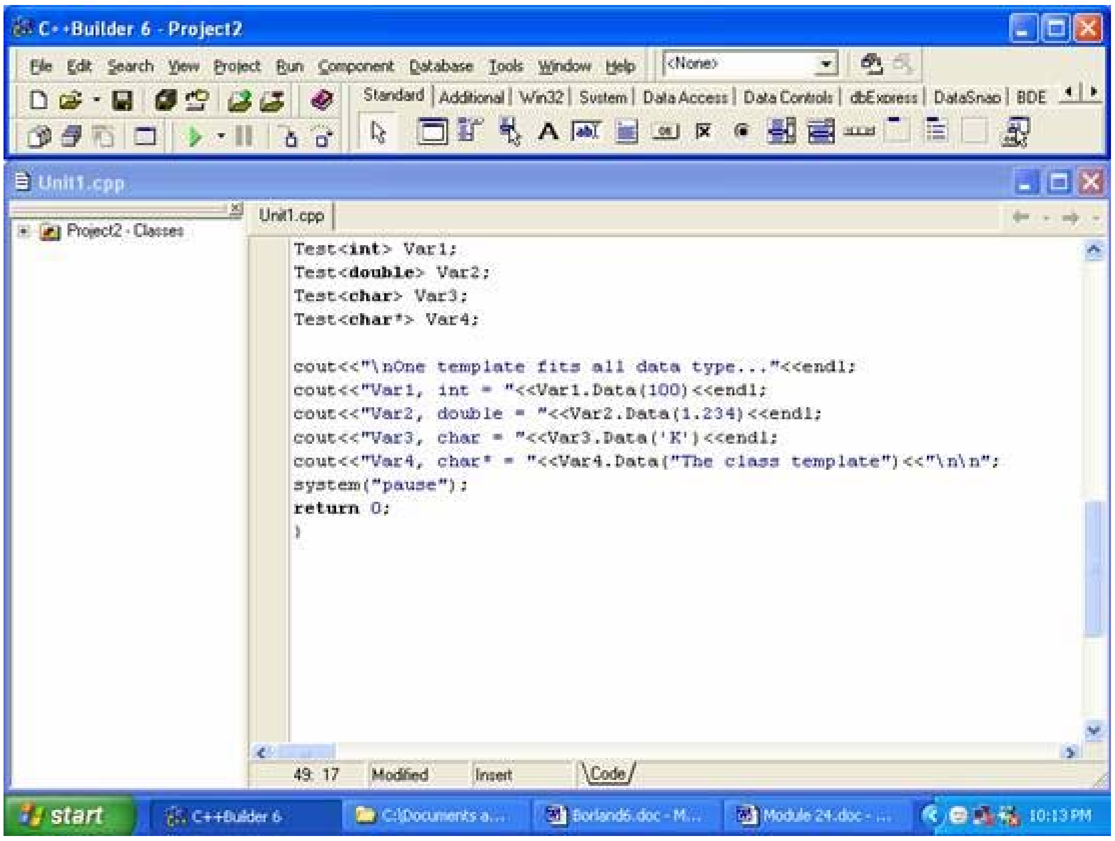
6.3 Borland C-

It is one type of Graphical User Interface.

When we load the program into the Rabbit processor and then remove the serial cable connection for testing of the system. We are going to assign Rabbit Module as receiver and Personal computer as Transmitter. Hence, after transmission when we get the acknowledgement, we can see that signal using this software named as Borland C.

Turbo C++ is a Borland C++ compiler with an integrated IDE. It was a part of Borland's highly popular family of compilers including Turbo Pascal, Turbo Basic, Turbo Prolog and Turbo C. Turbo C++ was a successor of Turbo C, expanding the compiler similarly to how Turbo Pascal 5.5 added object functionality to the earlier Turbo Pascal versions. Unlike Turbo Pascal, however, Turbo C++ always adhered to then-current C++ language standards. To newest version can be downloaded at the external link below.

Turbo C++ 3.0 was released in 1992, and came in amidst expectations of the coming release of Microsoft Windows 3.1. Turbo C++ v3.0 first came as an MS-DOS compiler, supporting C++ templates, generation of DOS & protected mode executables, as well as generation of code targeting specific legacy CPUs, such as Intel 80186. Turbo C++ brand was succeeded by the more robust Borland C++, the latter losing Turbo's main success features, such as speed and highly-convenient IDE, while adding up to date ANSI/ISO C++ conformance and more Windows programming functionalities. Borland C++ existed simultaneously with the RAD tool Borland C++ Builder for a while, until the latter took over completely, when RAD tools became more and more popular.



```
Test<int> Var1;
Test<double> Var2;
Test<char> Var3;
Test<char*> Var4;

cout<<"\nOne template fits all data type..."<<endl;
cout<<"Var1, int = "<<Var1.Data(100)<<endl;
cout<<"Var2, double = "<<Var2.Data(1.234)<<endl;
cout<<"Var3, char = "<<Var3.Data('K')<<endl;
cout<<"Var4, char* = "<<Var4.Data("The class template")<<"\n\n";
system("pause");
return 0;
}
```

Figure 34 Borland C look

CHAPTER 7 – SYSTEM PERFORMANCE EVALUATION

Receiver Section

7.1 Algorithm-

7.1.1 Main Program-

```
I.     Start
II.    Initialize message array, int I, key
III.   Initialize send response=false
IV.    Set 'for loop' to initialize the message array with NULL pointers
V.     Define messages like key [0] ="I got it", key [1] ="thanks"
VI.    Initialize the total system
VII.   Initialize the TCP sockets
VIII.  Allocate messages to each keypad
IX.    Set while loop while true
X.     Execute function keyprocess
XI.    Check whether message is received?
XII.   If no, then wait for the incoming message
XIII.  If yes, then press any key to proceed
XIV.   Check whether key is pressed?
XV.    If no, then wait until key is pressed
XVI.   If yes, then call function send message
XVII.  Set send respose=true
XVIII. End
```

7.1.2 Functions-

System Initialize-

I. Start

II. Initialize board

III. Initialize LCD

IV. Initialize keypad

V. Turn on the backlight

Receive message-

I. Start

II. Create buffer of size 500

III. Create integer numbyte

IV. Call the function tcp_listen

V. Clear the screen

VI. Display “Message received”

VII. Show displayed message

7.2 Flowchart

7.2.1 Main Program –

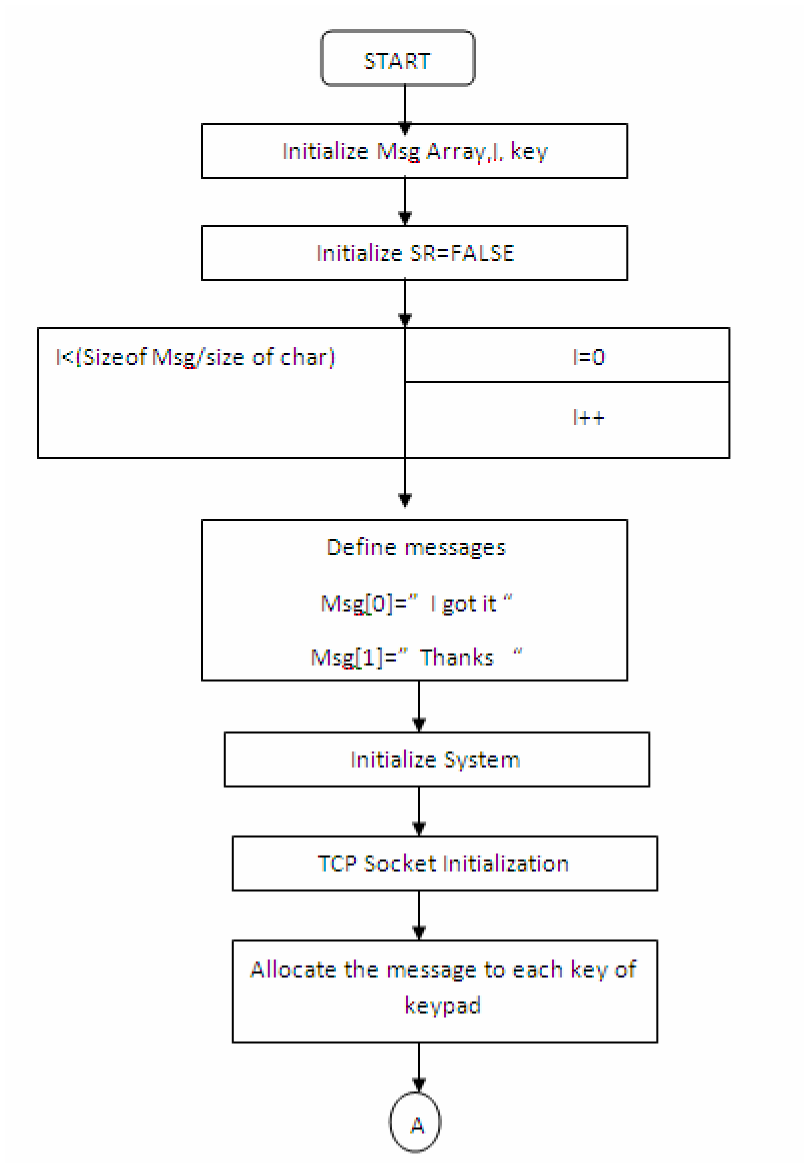

START
Initialize Msg Array,I, key
Initialize SR=FALSE
I<(Sizeof Msg/size of char)
I=0
I++
Define messages
Msg[0]=" I got it "
Msg[1]=" Thanks "
Initialize System
TCP Socket Initialization
Allocate the message to each key of keypad
A

Figure 35 Main Program Flow Rx I

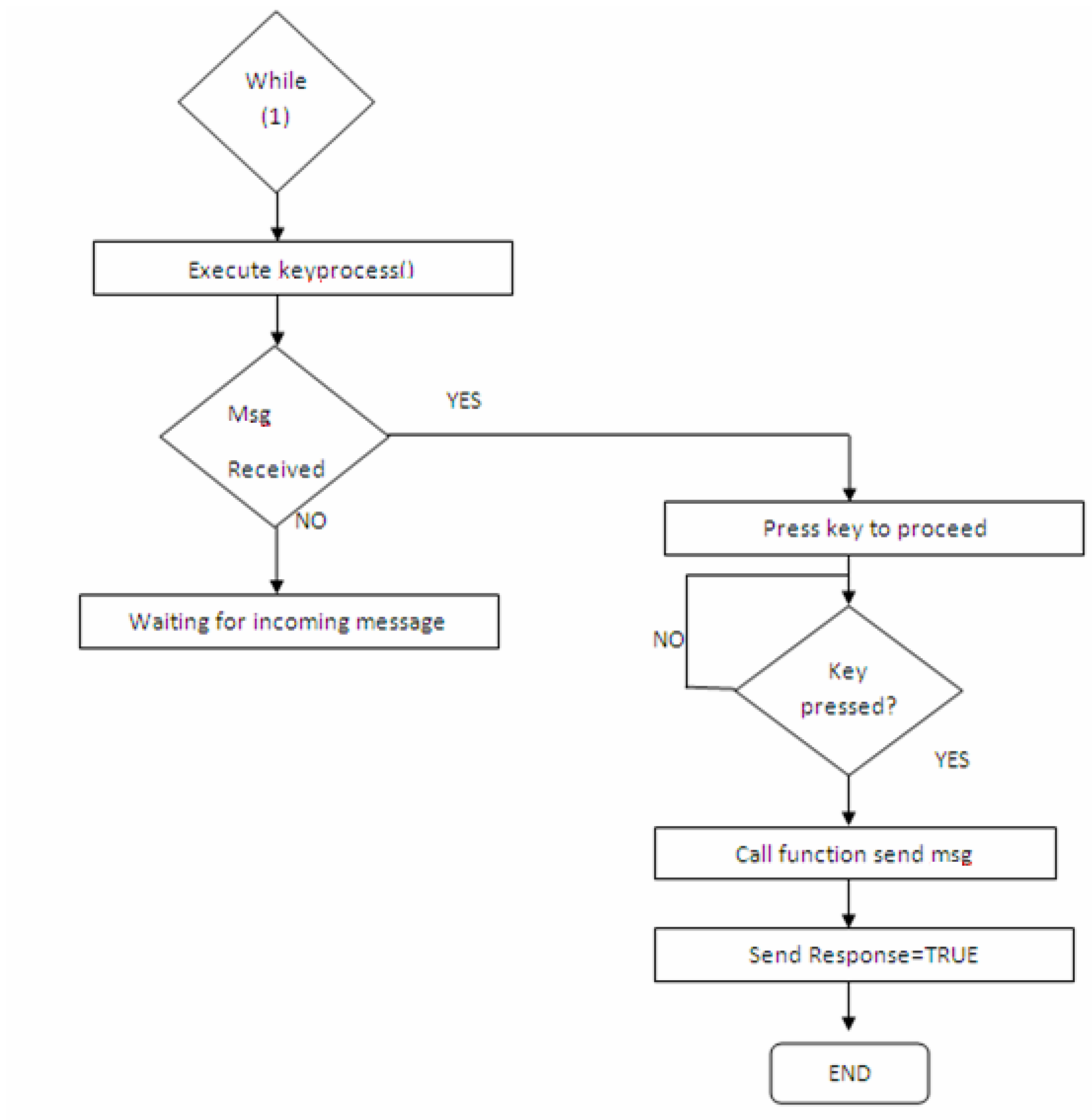


Figure 36 Main Flow Rx II

7.2.2 Functions-

System Initialize

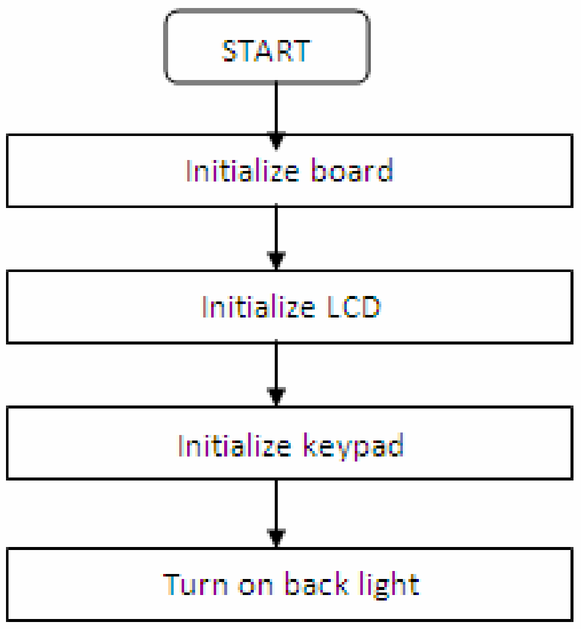


Figure 373 Subroutine Rx [System Initialize]

Receive Message-

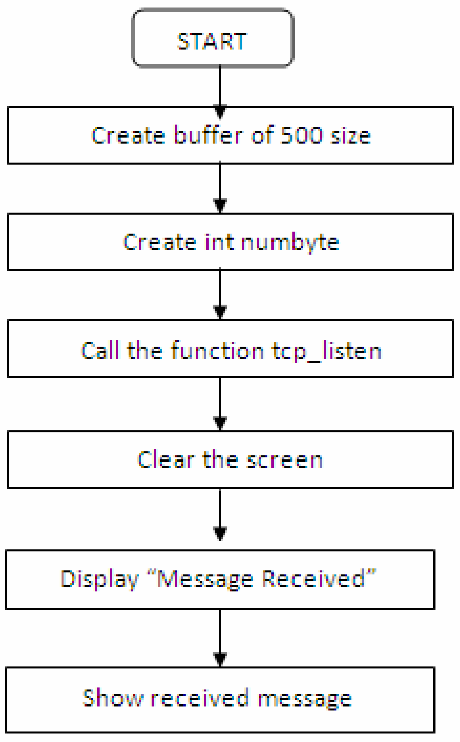

START
Create buffer of 500 size
Create int numbyte
Call the function tcp_listen
Clear the screen
Display "Message Received"
Show received message

Figure 38 Subroutine Rx [Receive Message]

Transmitter Section

7.3 Algorithm-

I. START
II. Include necessary header files.
III. Define IP address And Port.
IV. Declare WORD wVersion Requested, WSADATA wsaData, Int err, SOCKET sock, SOCKADDR_IN sin.

V. Define character length and message.
VI. String copy used to copy the IPADDR.
VII. String copy used to copy the PORT
VIII. wVersion Requested is assigned the version of windows socket 1.1.
IX. Socket initialize information and socket version is assigned to err variable using WSAStartup

X. If err!=0
XI. If Yes, Print (“WSAStartup() failed!”)
XII. If NO, sock = socket (PF_INET, SOCK_STREAM, 0);

XIII. If sock==INVALID_SOCKET
XIV. If Yes, Print (“socket () failed”)
XV. If No, sin.sin_family = AF_INET; sin.sin_addr.s_addr = 0; sin.sin_port = htons (port);
XVI. Establish connection

XVII. If (err==SOCKET_ERROR)
XVIII. If Yes, Print (“connection () failed”)
XIX. If No, Send message

XX. If err==SOCKET_ERROR
XXI. If Yes, Print ("send () failed!")
XXII. If No, Receive message

```
XXIII.   If err==SOCKET_ERROR
XXIV.    If Yes, Print ("recv () failed!")
XXV.     If No, Data [err] = null

XXVI.    Print ("data")
XXVII.   Close socket
XXVIII.  If err==SOCKET_ERROR
XXIX.    If Yes, Print ("closesocket () failed!")
XXX.     If No, WSACleanup(), return 0
XXXI.    STOP
```

7.4 Flowchart-

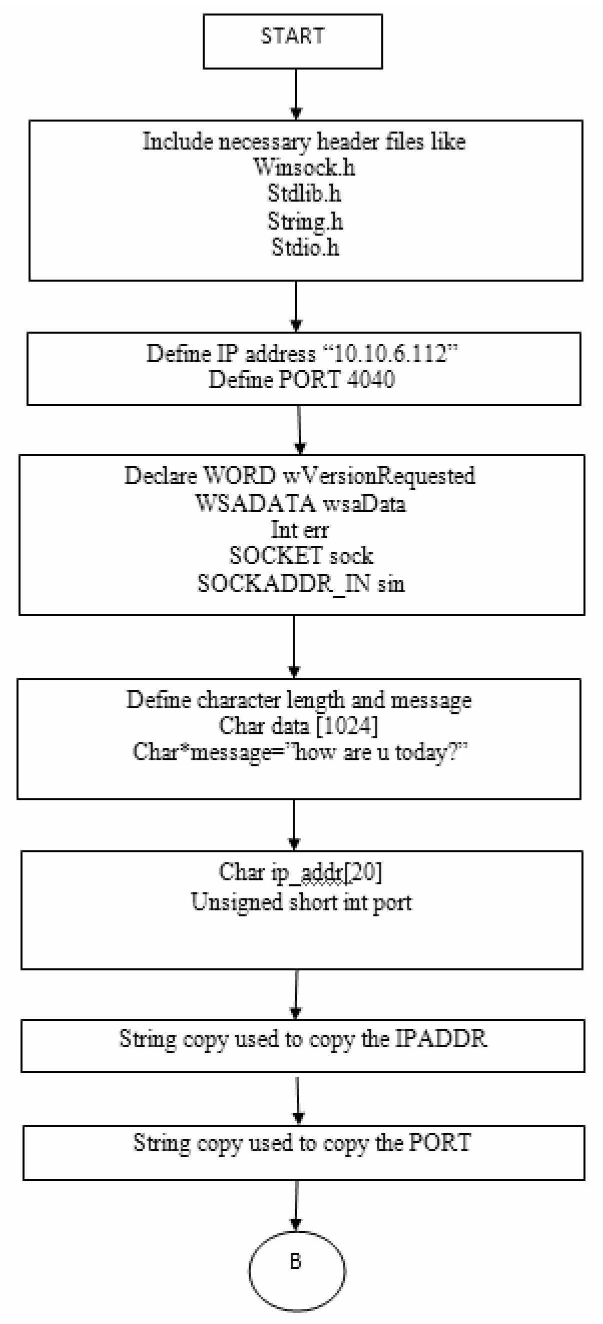

START
Include necessary header files like
Winsock.h
Stdlib.h
String.h
Stdio.h
Define IP address "10.10.6.112"
Define PORT 4040
Declare WORD wVersionRequested
WSADATA wsaData
Int err
SOCKET sock
SOCKADDR_IN sin
Define character length and message
Char data [1024]
Char*message="how are u today?"
Char ip_addr[20]
Unsigned short int port
String copy used to copy the IPADDR
String copy used to copy the PORT
B

Figure 39 Flowchart Tx I

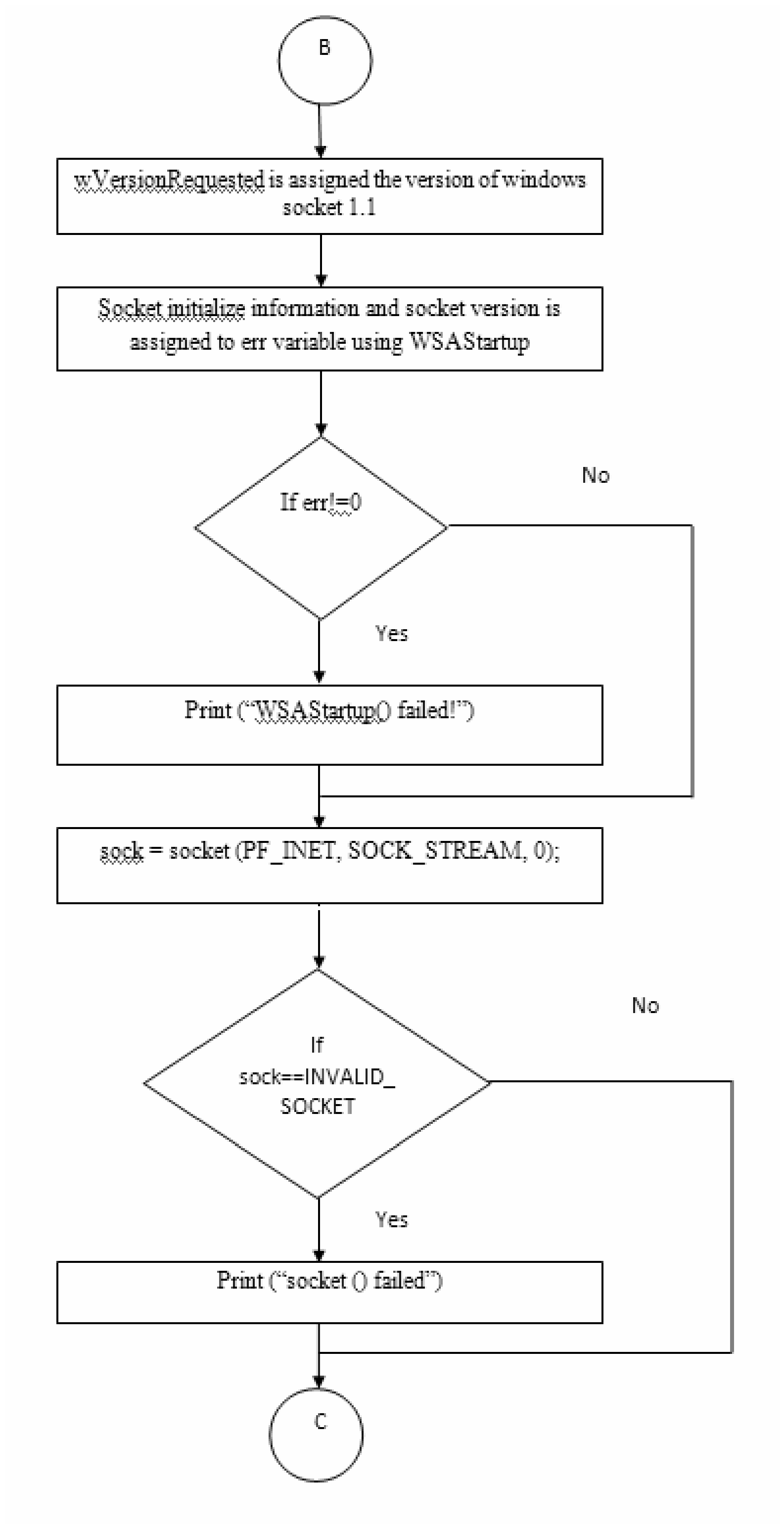

B
wVersionRequested is assigned the version of windows socket 1.1
Socket initialize information and socket version is assigned to err variable using WSAStartup
If err!=0
No
Yes
Print ("WSAStartup() failed!")
sock = socket (PF_INET, SOCK_STREAM, 0);
If sock==INVALID_ SOCKET
No
Yes
Print ("socket () failed")
C

Figure 40 Flowchart Tx II

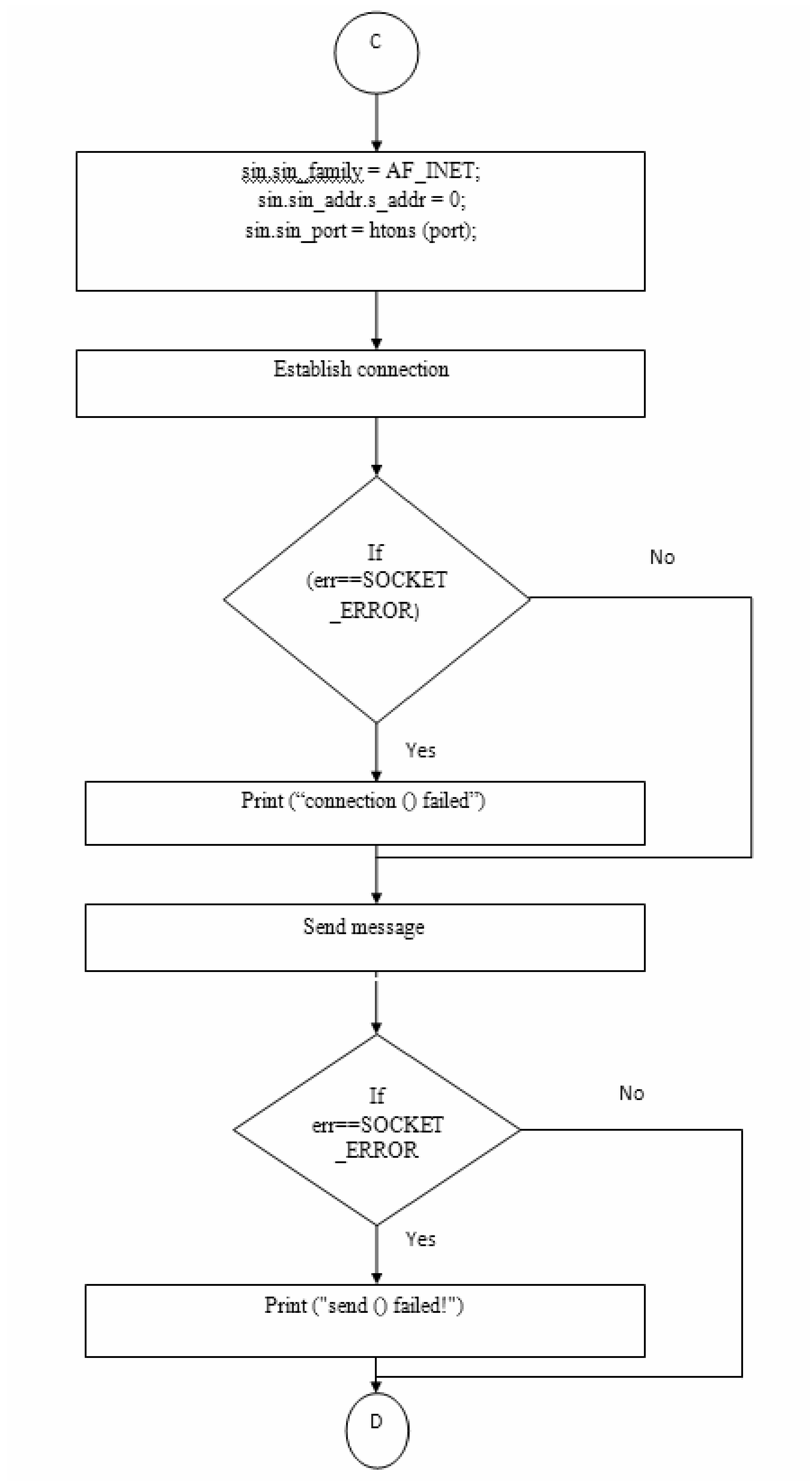


Figure 41 Flowchart Tx III

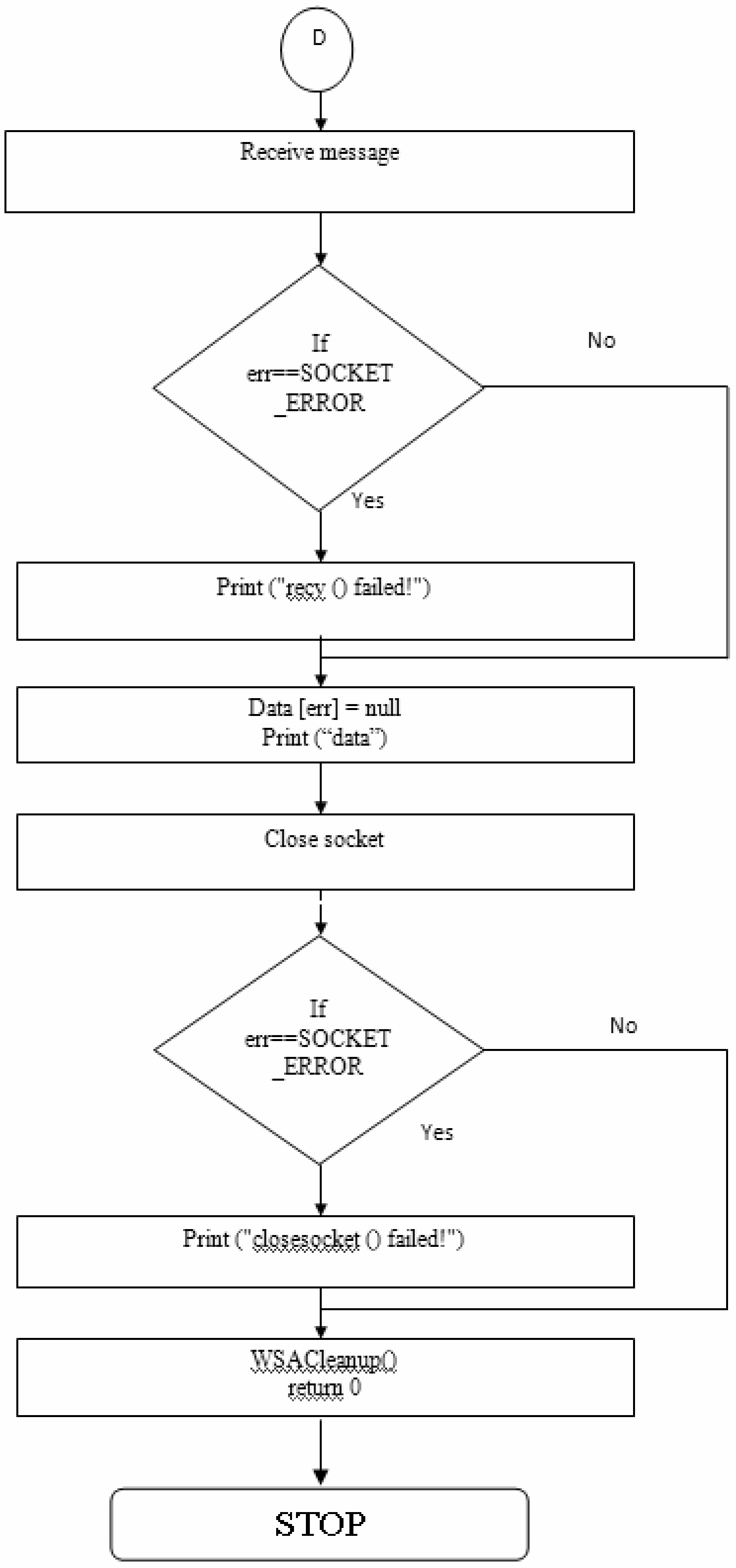


Figure 42 Flowchart Tx IV

7.5 Problems faced and their solutions-

We came across many problems while designing this system, some of the problems we have mentioned here with their solution.

Problem -
When we had been told to design this system using Rabbit processor, we were newly introduced to the world of Rabbit. Then the question came, what is Rabbit processor?
Solution-
We took a survey of Rabbit Processors and their features. We studied the Rabbit 4000 User manual to know about Rabbit processor.

Problem -
How to program Rabbit processor? What is Dynamic C?
Solution-
Dynamic C is software provided to program Rabbit processor. Whenever you purchase Rabbit kit then you will get this software.

Problem -
How to write any general program in Dynamic C? How to execute it on Rabbit Processor?
Solution-
Then we studied the Dynamic C function manual to know meaning and syntax of each and every function. Then we practiced some sample programs and tried to execute them on Rabbit kit.

Problem -
We only knew theory of TCP, but how to implement on Rabbit processor?
Solution-
We studied TCP from the book of Behrouz A Forouzan,"Data communication and Networking", McGraw Hill Publication and studied how to implement it on Rabbit processor from Jan Alexson,"Embedded Ethernet and Internet Complete Designing", Lakeview Research, 2003 book.

Problem -

What are the steps to be followed while implementing the TCP stack on Rabbit processor?
Solution-
Then we used book Kamal Hyder & Bob Perria ,"Embedded Systems Design using the Rabbit 3000 Microprocessor interfacing, networking and application development", ISBN: 0-7506-7872-0, 2005 to see the procedure of implementing TCP stack.

Problem -

After finishing all the programs, we were ready to execute them on Rabbit Kit. But the necessary library files for them were missing. Hence, how to proceed?

Solution-

We searched on various forums site and posted queries related to non availability of library files of Rabbit Processor. Mainly we posted the posts on the official site of Rabbit Semiconductor, www.forums.digi.com.We got help related to library files from various people all around the globe.

Problem -

After getting all the required library files, how to use them in the program and edit them as per requirement?

Solution-

Then also we posted queries on the forums while editing the library files as per the requirement. We got the help from forums only.

Problem-

How to find the ports for hardware interfacing?

Solution-

We searched into all the required library files to finds the ports for connection of Data and Control Signals.

CHAPTER 8 – FUTURE SCOPE

8.1 Future Expansion and Scope-

To enhance the data transfer rate some modifications can be developed in the project, they are as follows-

I. We can change the cable type, i.e. instead of using CAT5 cable we can use cables with fast data transfer rates like fiber optic cables.

II. We can change to enhance the data transfer rate, RCM 4300 supports 10Mbps speed, we can use processor which supports high data rate e.g. RCM 5000.

III. Personal computer which we are using has Pentium 4 processor inside, but we can use processors like core 2 duo, core i3, core i5, etc to have very fast transfer of the instructions to Rabbit.

CHAPTER 9 - SYSTEM COSTING

9.1 List of Hardware Material required and Cost-

SR	Name of component	Required Pieces	Cost/Unit	Total cost
1]	Rabbit 4000 module With development Board	1	Rs.30000/-	Rs.30000/-
2]	Personal Computer	1	-	Rs.15000/-
3]	Ethernet cable [CAT5]	5m	Rs 18/m	Rs.90/-
4]	LCD 132x32	1	Rs 1000/-	Rs.1000/-
			Total cost	**Rs. 46090/-**

Table 6 System Costing

CHAPTER 10 - CONCLUSION

The full featured TCP/IP stack from Rabbit along with the PC containing Dynamic C, allows a system designer to design with a minimum investment in time and money. As with any technology, tools used to mold the technology into products are as important as the technology. Rabbit's highly integrated embedded processor is remarkable technology. Also when it is used in the system, the system gives better performance than µc8051 and µp8086 traditional processors in terms of data rate, networking, ease of programming, and hardware interface. Hence it is proved that Ethernet based communication between PC and Rabbit Processor provides better performance.

REFERENCES

1) Dynamic C Functional Reference Manual.

2) Jan Alexson ,"Embedded Ethernet and Internet Complete Designing", Lakeview Research, 2003

3) Kamal Hyder & Bob Perria ,"Embedded Systems Design using the Rabbit 3000 Microprocessor interfacing, networking and application development", ISBN: 0-7506-7872-0, 2005, Elsevier Inc

4) Behrouz A Forouzan ,"Data Communications and Networking", Third Edition, McGraw Hill Publication

5) Rabbit 4000 User Manual

6) Dynamic C TCP/IP User's Manual.

7) A two part article, Introduction to TCP/IP in Embedded Systems programming discusses related to programming embedded systems. ejsr_22_2_15.pdf

8) GMRT.pdf

9) W.Richard Stevens, "TCP/IP Illustrated Volume 1 The Protocols", Addison Wesley Publication, ISBN 0-20-163346-9

PUBLICATIONS

I. Nihar S Bendre, Pramesh M Sabne, Amey A Kulkarni, "Ethernet Based Communication Between Rabbit Processor & Personal Computer ", National Conference on Communication and Pattern Recognition (NCONCPR 2012), Pune, Paper ID- NCCM 046, Article 104, Pp45.

II. Nihar S Bendre, Pramesh M Sabne, Amey A Kulkarni, "Ethernet Based Communication Between Rabbit Processor & Personal Computer ", 2nd National Conference on Communication Network and Sensor Technology, Hindustan University, Chennai, Paper ID- SEN126, Pp 304-308, ISBN- 978-81-89843-50-2. .

III. Nihar S Bendre, Pramesh M Sabne, Amey A Kulkarni, "Ethernet Based Communication Between Rabbit Processor & Personal Computer ", International Journal of Science and Engineering Research [IJSER], France, Paper ID- I014989 ,Volume- 3, Issue- 5,May 2012, ISSN- 2229-5518.